# UNIVERSIDAD POLITÉCNICA SALESIANA

## CARRERA DE INGENIERÍA DE SISTEMAS

### MENCIÓN TELEMÁTICA

### TÍTULO O TEMA DE TRABAJO

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA SUPERVISIÓN DE SENSORES APLICADOS A LA DOMÓTICA CON COMUNICACIONES EN REDES DE ÁREA PERSONAL ENTRE DISPOSITIVOS MÓVILES CON CONTROL EN TIEMPO REAL MEDIANTE FPGA.

**TESIS PREVIA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS**

**AUTORES**

**DARWIN JOSÉ CASTILLO CASTILLO**

**RICHARD EDUARDO OSORIO SALCEDO**

**DIRECTOR**

**ING. RAFAEL JAYA**

**Quito, Junio 2011**

# DECLARACIÓN

Nosotros, Darwin José Castillo Castillo, Richard Eduardo Osorio Salcedo, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondiente a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la ley de Propiedad Intelectual, por su reglamento y su normativa institucional vigente.

| | |
|---|---|
| _____ | _____ |
| **Darwin José Castillo Castillo** | **Richard Eduardo Osorio Salcedo** |

# CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Darwin José Castillo Castillo, Richard Eduardo Osorio bajo mi dirección.

_____

**Ing. Rafael Jaya**

**Director de Tesis**

# DEDICATORIA

La presente tesis está dedicada a mis padres, hermanas y abuelitas, pilares fundamentales en mi vida, sin ellos, jamás hubiese podido conseguir lo que hasta ahora. Su tenacidad y lucha insaciable han hecho de ellos el gran ejemplo a seguir y destacar.

También dedico este proyecto a mi novia, compañera inseparable de cada jornada quien constantemente me impulsaba a seguir adelante día a día.

**Richard Eduardo Osorio Salcedo**

# AGRADECIMIENTO

A cada uno de los que son parte de mi familia a mi padre Eduardo Osorio, mi madre Yolanda Salcedo, mis hermanas Mónica, Maricela, Jenny y toda mi familia quien siempre estuvo a mi lado.

A mi Ivonne, por siempre haberme dado su fuerza y apoyo incondicional que me han ayudado y llevado hasta donde estoy ahora.

Al director de tesis quién nos ayudó en todo momento a través de sus tutorías, Ing. Rafael Jaya.

A Darwin Castillo que supo ser un buen compañero durante el proceso de desarrollo de tesis.

**Richard Eduardo Osorio Salcedo**

# DEDICATORIA

Agradezco a mi tío Sergio Castillo que ha sido la persona más importante de mi vida, gracias a su esfuerzo y sacrificio, el se ha preocupado por mi educación personal y académica para que sea un profesional con grandes principios y aspiraciones en la vida, en reemplazo de mi padre Segundo Castillo que desde su partida, él esta junto a mi en los momentos tristes y alegres de mi vida.

También dedico mi tesis a mi compañero Richard Osorio que siempre ha estado apoyándome en el éxito de la superación profesional como personal a mis tíos EFREN ,LIVIO, VICENTE, BENILDA, BENITO, ROSENDO, OLGA, RAUL , mis primos MAURICIO, LEODAN, ALCIVAR, FRANKLIN, ROSENDO, VICENTE ROBERT JOSE y a mis primas GLADYS,Y MAGDALENA SONIA y VERONICA los cuales han vivido mi juventud siempre estamos incentivándonos del uno a otro para lograr que seamos buenas personas de bien y buenos profesionales ante la sociedad.

Al igual que a mi familia a mi amigo MARCO CAZAR y a ELILIANA CALVA que ha sido la persona que ha estado en los momentos más difíciles de mi vida siempre han estado brindándome un apoyo moral y psicológico, de no dejarse vencer por las adversidades de la vida porque los problemas no son un obstáculo de la vida si no una experiencia más.

ANITA TORRES te dedico mi tesis por que eres la persona que siempre se ha preocupado por bienestar tanto de mí como de mi familia nunca olvidare el sacrificio y esfuerzo que sigues haciendo por los demás.

**Darwin José Castillo Castillo**

# PRESENTACIÓN

**CAPITULO I**

Plan de tesis

**CAPITULO II**

Marco Teórico

**CAPITULO III**

Análisis y Requerimientos

**CAPITULO IV**

Pruebas y Resultados

# RESUMEN

El objetivo de esta tesis, se centra en establecer comunicación dentro de una red de área personal PAN mediante un FPGA (Field Programable Gate Array – Arreglo de compuertas de campo programable) y dispositivos inalámbricos.

Se eligió Zigbee ya que es un conjunto de protocolos de alto nivel de comunicación inalámbrica, debido a que es una red de bajo costo que no necesita una previa conexión hacia internet para su funcionamiento, facilitando el envío de datos.

Para la descripción del hardware, se utilizó el lenguaje VHDL (Very Hardware Description Lenguaje), ésta programación es cargada dentro del entrenador FPGA, la cual permite establecer una óptima comunicación con los dispositivos inalámbricos como medio de transmisión.

Al tener comunicación inalámbrica dentro de una red PAN se puede controlar sensores y actuadores mediante una interfaz, los sensores utilizados son: un sensor de temperatura, de humo y de movimiento los mismos que envían datos constantemente hacia la interfaz, siendo el manejo amigable para el usuario, el cual puede escoger el número de puerto (COM) gráficamente.

# ÍNDICE GENERAL

**CAPÍTULO III**

**CAPÍTULO IV**

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# CAPITULO I

# 1. PLAN DE TESIS

## TEMA:

**ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA SUPERVISIÓN DE SENSORES APLICADOS A LA DOMÓTICA CON COMUNICACIONES EN REDES DE ÁREA PERSONAL ENTRE DISPOSITIVOS MÓVILES CON CONTROL EN TIEMPO REAL MEDIANTE FPGA.**

## 1.1 ANTECEDENTES

La tecnología FPGA utiliza procesos de forma simultánea para que no exista encolamiento de información y los procesos actúen de forma real.

Las FPGAs se utilizan en aplicaciones similares a los ASICs sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Ciertos fabricantes cuentan con FPGAs que sólo se pueden programar una vez, por lo que sus ventajas e inconvenientes se encuentran a medio camino entre los ASICs y las FPGAs reprogramables.

Históricamente las FPGA surgen como una evolución de los conceptos desarrollados en las PAL y los CPLD.

Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema, algo parecido a un breadboard (es una placa de uso genérico reutilizable o semi permanente) programable. Estos bloques lógicos e interconexiones pueden ser programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un «Sistema programable en un chip». Ejemplo de tales tecnologías híbridas pueden ser encontradas en los dispositivos Virtex-II PRO y Virtex-4 de Xilinx, los cuales incluyen uno o más procesadores PowerPC embebidos junto con la lógica del FPGA. El FPSLIC de Atmel es otro dispositivo similar, el cual usa un procesador AVR en combinación con la arquitectura lógica programable de Atmel. Otra alternativa es hacer uso de núcleos de procesadores implementados haciendo uso de la lógica del FPGA. Esos núcleos incluyen los procesadores MicroBlaze y PicoBlaze de Xlinx, Nios y Nios II de Altera, y los procesadores de código abierto LatticeMicro32 y LatticeMicro8.

Muchos FPGA modernos soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada, mientras las demás partes siguen funcionando. Este es el principio de la idea de la (computación reconfigurable), o los (sistemas reconfigurables).

La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL o Hardware Description Language (lenguajes de descripción de hardware). Los HDLs más utilizados son:

- VHDL
- Verilog
- ABEL

En un intento de reducir la complejidad y el tiempo de desarrollo en fases de prototipaje rápido, y para validar un diseño en HDL, existen varias propuestas y niveles de abstracción del diseño. Entre otras, National Instruments LabVIEW FPGA propone un acercamiento de programación gráfica de alto nivel.

## 1.2 Planteamiento del Problema

Actualmente existen dispositivos electrónicos de alto costo para poder controlar sensores y actuadores en el campo de la domótica mediante cables o cámaras, es por eso que se va a diseñar una aplicación para controlar los sensores y actuadores mediante un dispositivo electrónico de banda base, lo cual brindará seguridad en hogares o empresas sin tener que estar en dicho sitio mediante redes de área personal (PAN).

Anteriormente para las comunicaciones se utilizaban dispositivos electrónicos analógicos, los cuales para realizar un proceso se almacenaba primeramente la información y posteriormente se ejecuta las transmisiones para realizar la acción o tarea, con el uso de la tecnología FPGA lo que se busca es que la información se haga simultáneamente sin que la información se encole y muestre las tareas o procesos en forma real.

Las comunicaciones dentro de una PAN especialmente dentro de la domótica, se destina para reemplazar los cables de conexión portátil y dispositivos electrónicos para la transferencia de información.

Con los mismos, se podrá controlar secuencias de encendido y apagado de los sensores y actuadores dentro de la domótica a través de lenguaje de programación VHDL.

Dentro de la domótica el poder controlar los sensores y actuadores en tiempo real se hace necesario pero los actuales microcontroladores no ofrecen esta característica, por lo tanto el avance de las nuevas tecnologías ofrece la arquitectura FPGA (Field Programmable Gate Array) la misma que con la lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

## 1.3 OBJETIVOS

### 1.3.1 Objetivo General

Diseñar e implementar una aplicación para establecer radio comunicación en redes de área personal entre dispositivos móviles, mediante FPGA.

### 1.3.1 Objetivos Específicos

- Realizar un estudio sobre las radiocomunicaciones para redes PAN en dispositivos móviles.
- Diseñar e implementar un circuito para establecer una comunicación bidireccional entre dispositivos móviles mediante tecnología FPGA.
- Diseñar e implementar un circuito para realizar transferencia de información con redes PAN utilizando la tecnología FPGA para dispositivos móviles.
- Utilizar terminales de comunicación entre en circuito y los dispositivos móviles.
- Utilizar un FPGA para una aplicación en el campo de la comunicación entre dispositivos inalámbricos.
- Realizar una aplicación la cual verificará el funcionamiento de los sensores y actuadores en el área de la domótica.
- Diseñar un prototipo el cual va a contener los sensores y actuadores a controlar mediante la tecnología FPGA.
- Controlar mediante un FPGA los estados de los sensores y actuadores en el área de la domótica

## 1.4 JUSTIFICACIÓN DEL PROYECTO

El desarrollo de la tesis se basa específicamente en la integración y configuración de elementos los cuales hacen posible establecer la comunicación bidireccional entre dispositivos (MOVILES Y FPGA).

Los FPGAs son dispositivos que combinan muchas de las novedades en el diseño de circuitos integrados para la implementación de sistemas digitales, y brindan la posibilidad de ajustarse a necesidades individuales, definidas por el usuario. Son circuitos lógicos programables muy poderosos, se puede diseñar sistemas completos en la computadora utilizando un lenguaje de descripción apropiado.

Las nuevas tecnologías FPGA aplicadas en la domótica poseen un gran futuro al poder establecer comunicación bidireccional con dispositivos móviles.

La necesidad de poder controlar sensores y actuadores dentro de la domótica a través de dispositivos móviles con tecnología FPGA, y poder unirlos a redes PAN es de mucha utilidad para controlar procesos simultáneos.

## 1.5 DESCRIPCIÓN GENERAL DEL PROYECTO

A continuación, se realizará una descripción general del proyecto, con el fin de enmarcarlo en la realidad socioeconómica de nuestro país. En el presente proyecto realizará el análisis diseño e implementación de un prototipo para la supervisión de sensores aplicados a la domótica con comunicaciones en redes de área personal entre dispositivos móviles con control en tiempo real mediante FPGA.

Para lo cual, se utilizará un lenguaje de programación denominado VHDL para el correcto funcionamiento de tecnologías FPGA de acuerdo a las necesidades del usuario.

Además el presente proyecto pretende mostrar una manera de ahorrar la energía eléctrica utilizada para la iluminación de un hogar o de una oficina, administrando su uso de acuerdo a la demanda, no sólo en algo tan básico como encenderla sólo cuando va a ser utilizada y apagarla en caso contrario, sino en algo más elaborado como regular su intensidad de acuerdo a la necesidad y lograr realizarlo por sectores y de manera remota utilizando dispositivos móviles aplicados a redes de área personal (PAN).

# CAPÍTULO II

## 2. SUSTENTO TEÓRICO

En este capítulo, se proporciona las bases teóricas que sustentan el proyecto de tesis, tecnologías, dispositivos, módulos, utilizados para el posterior análisis desarrollo implementación del proyecto.

## 2.1 PLATAFORMA FPGA

### 2.1.1 Antecedentes de FPGA

Un diseñador de sistemas electrónicos dispone de diversas opciones para implementar la lógica digital, incluyendo dispositivos lógicos discretos frecuentemente llamados Circuitos integrados de pequeña escala (SSI); dispositivos programables tales como Arreglos de lógica programable (PALs o PLDs); Arreglos de compuertas programadas; y Arreglos de compuertas programables en el campo (FPGA´s) ver figura [1]. Un FPGA[1] es un dispositivo multinivel programable de propósito general. Integra una gran cantidad de dispositivos lógicos programables en un chip. El tamaño y velocidad de los FPGA´s es equiparable a los Circuitos Integrados para Aplicaciones Específicas (ASICs)[2], pero los FPGA´s son más flexibles y su ciclo de diseño es más corto.

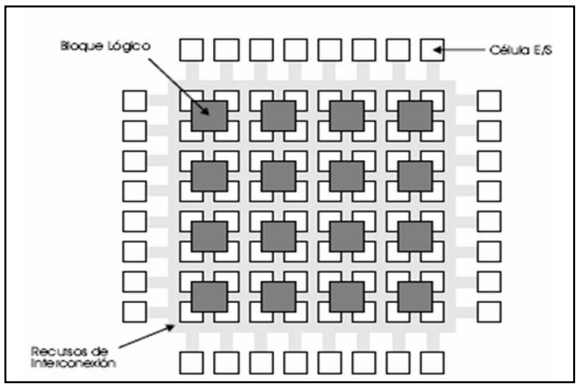



**Figura [1], Arquitectura FPGA**
**Fuente:** http://polibits.gelbukh.com/37_11.pdf

[1] Field Programmable Gate Array
[2] Circuito Integrado para Aplicaciones Específicas.

## 2.2 DESCRIPCIÓN FPGA

Las FPGA son dispositivos que permiten diseñar sistemas digitales para aplicaciones específicas, es el usuario quién finalmente decide en que se convertirá el dispositivo mediante su configuración.

Estos dispositivos poseen la ventaja de tener un puerto JTAG[3] Ver Figura [2] el cual es un estándar de la IEEE[4] estándar # 1149.1- 1990 que permite tanto la configuración del circuito integrado como la posibilidad de tener su funcionamiento en el circuito impreso sin necesidad de utilizar equipo de mediciones, permitiendo observar en tiempo real, las entradas, los procesos y las salidas en el dispositivo. Con lo cual se obtiene la opción de determinar rápidamente la depuración de los diseños, si el problema está en el diseño digital y/ó en la tarjeta del circuito impreso PCB[5].

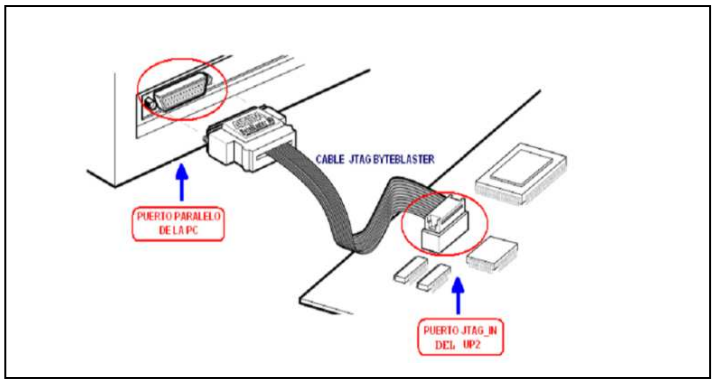



**Figura [2], Cable JTAG**
**Fuente:** http://www.zero13wireless.net/foro/showthread.php?632-JTAG-y-cables

Un FPGA consiste en un arreglo o matriz bidimensional de bloques configurables, que se pueden conectar mediante recursos generales de interconexión, que incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. Dicho de otra forma un FPGA es un arreglo

[3] Un acrónimo para Joint Test Action Group
[4] Institute of Electrical and Electronic Engineers
[5] Printer Circuit Boarb

de LBs[6] configurables, colocados en un arreglo programable de interconexiones; es decir, los LBs, los IOBs, y las interconexiones entre éstos, pueden ser programadas por el usuario. El esquema básico se muestra en la Figura [3].

Lo que se programa en sí en un FPGA son los conmutadores, que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

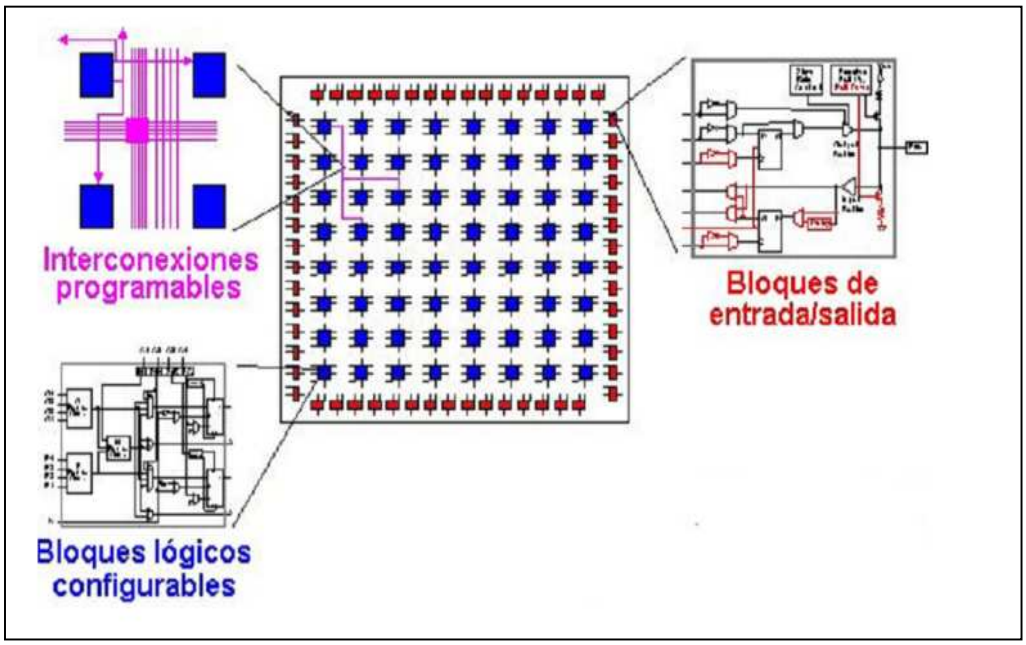



**Figura [3], Esquema básico de un FPGA**
**Fuente:** http://bifi.es/research/complexm_fundphysics/ssue/ssue.php

La comunicación entre la PC y el FPGA, donde se transfiere la información (bits), necesaria para realizar el proceso de configuración, se lo hace a través del puerto JTAG y con la ayuda de las herramientas de diseño.

Aunque existen diferentes arquitecturas para los FPGAs, dependiendo del fabricante el esquema básico consta de tres elementos que son: bloques lógicos configurables (CLBs),

[6] location-based service

interconexiones programables y bloques de entrada y/o salida (IOBs). Estos elementos se muestran en la Figura [3].

## 2.3 EVOLUCIÓN DE LA TECNOLOGÍA FPGA

Las FPGAs fueron inventadas en el año 1984 por Ross Freeman y Bernard Vonderschmitt, co-fundadores de Xilinx, y surgen como una evolución de los CPLDs[7].

Tanto los CPLDs como las FPGAs contienen un gran número de elementos lógicos programables. Si medimos la densidad de los elementos lógicos programables en puertas lógicas equivalentes (número de puertas NAND equivalentes que podríamos programar en un dispositivo) podríamos decir que en un CPLD hallaríamos del orden de decenas de miles de puertas lógicas equivalentes y en una FPGA del orden de cientos de miles hasta millones de ellas.

Aparte de las diferencias en densidad entre ambos tipos de dispositivos, la diferencia fundamental entre las FPGAs y los CPLDs es su arquitectura. La arquitectura de los CPLDs es más rígida y consiste en una o más sumas de productos programables cuyos resultados van a parar a un número reducido de biestables[8] síncronos (también denominados flip-flops). La arquitectura de las FPGAs, por otro lado, se basa en un gran número de pequeños bloques utilizados para reproducir sencillas operaciones lógicas, que cuentan a su vez con biestables síncronos. La enorme libertad disponible en la interconexión de dichos bloques confiere a las FPGAs una gran flexibilidad.

Otra diferencia importante entre FPGAs y CPLDs es que en la mayoría de las FPGAs se pueden encontrar funciones de alto nivel (como sumador y multiplicador) embebidas en la propia matriz de interconexiones, así como bloques de memoria.

[7] Un CPLD (del acrónimo inglés Complex Programmable Logic Device) es un dispositivo electrónico.
[8] Es un multivibrador capaz de permanecer en un estado determinado o en el contrario durante un tiempo indefinido.

## 2.4 ARQUITECTURA DE LA PLATAFORMA FPGA.

El FPGA que contiene la tarjeta es de la empresa Xilinx (XC3S500E-FG320), este contiene 500K compuertas que son equivalentes a 10476 celdas lógicas. Su arquitectura incluye 20 bloques de 18 Kb de RAM, 20 multiplicadores de hardware de 18 x 18 bits, 4 Digital Clock manangers y hasta 232 señales de E/S.

Los periféricos disponibles en la tarjeta son: Memoria Flash 16 MByte (128 Mbit) para aplicaciones, DDR (double data rate) SDRAM de 64 MByte (512 Mbit), CLPD XC2C64A, familia CoolRunner, Memoria Flash de 4 Mbit para configuración, Memoria Flash 16 Mbits acceso serial, vía SPI(serial peripheral interface), una interface de capa física Lan Ethernet 10/100 y un oscilador de 50 Mhz.

Los puertos externos que tiene la tarjeta son dos puertos seriales RS-232 de nueve terminales, un puerto VGA, un puerto PS/2 para teclado o mouse un puerto Ethernet 10/100 Mb/seg, dos puertos para la programación. El principal puerto de programación es un controlador empotrado USB. La tarjeta de desarrollo Spartan-3E tiene algunos accesorios.

Cuatro pulsadores, cuatro switchs, un botón rotatorio, ocho leds y una pantalla LCD de 16 caracteres por 2-líneas. Además tiene dos leds que verifican la alimentación y la configuración de la tarjeta. Tiene una conexión para expansión de 100 pines y tres conectores de 6 pines que se utilizan para ampliar la capacidad de la tarjeta adicionando periféricos externos por estos conectores.

La tarjeta integra un convertidor Digital a Analógico SPI de cuatro salidas (DAC), con resolución de 12 bits y un convertidor Analógico a Digital SPI de dos entradas.

En la figura [4] se observa el diagrama de bloques de la arquitectura básica implementada en la FPGA de SAKC. Esta arquitectura debe conservarse para permitir la comunicación entre el procesador y los periféricos implementados en la FPGA.

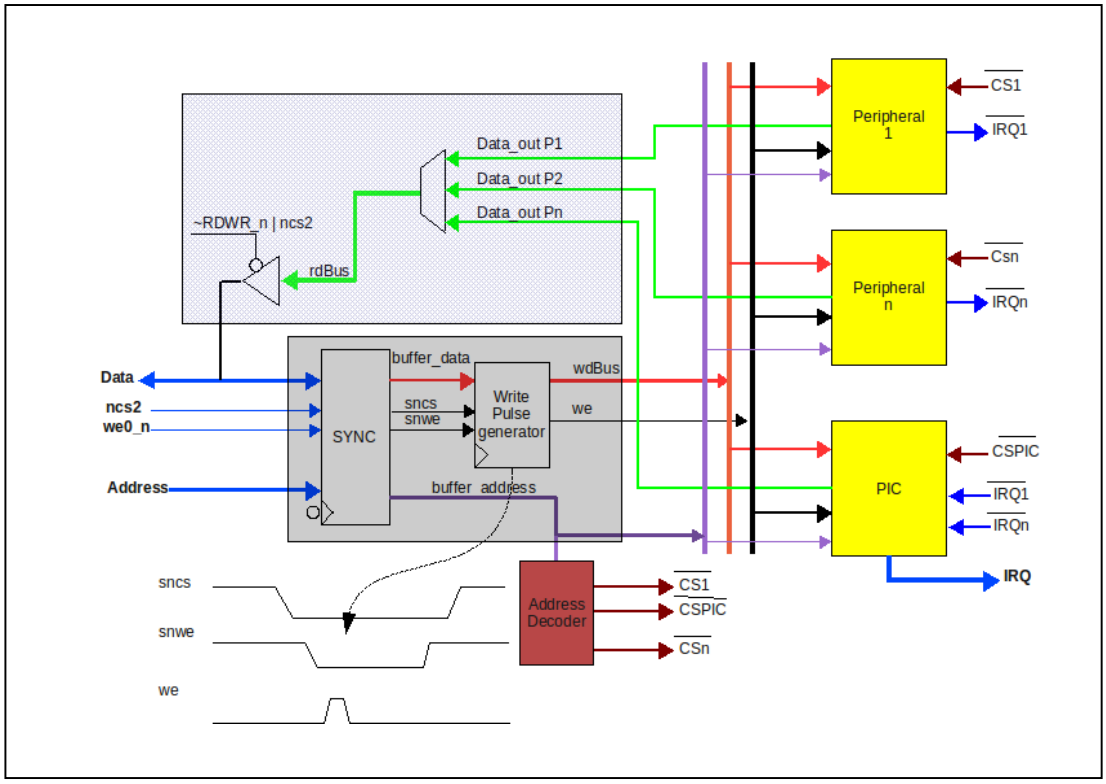



**Figura [4], Arquitectura básica del procesador implementada en la FPGA**
**Fuente:** http://redalyc.uaemex.mx/pdf/730/73012215014.pdf

Incluye memoria EPROM de configuración de la FPGA, 2 Mbytes de memoria RAM estática (SRAM), 4 Mbytes de memoria FLASH, un puerto serie, conector JTAG y conector de expansión de 200 pines.

La alta densidad de puertas, el gran número de pines entrada/ salida ( IOs) así como la memoria externa SRAM y FLASH, permiten al usuario, realizar la implementación de un diseño de bajo coste, en un tiempo record para obtener un producto final.

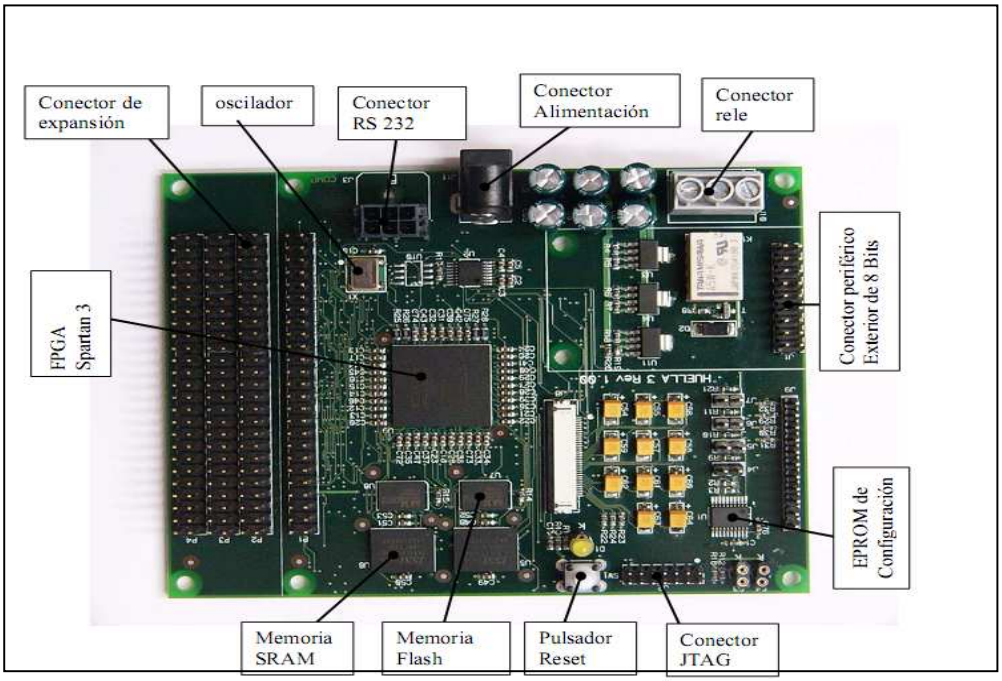



**Figura [5], Tarjeta de desarrollo y sus características**
**Fuente:** http://www.microafis.com/paginas/spartan3.pdf

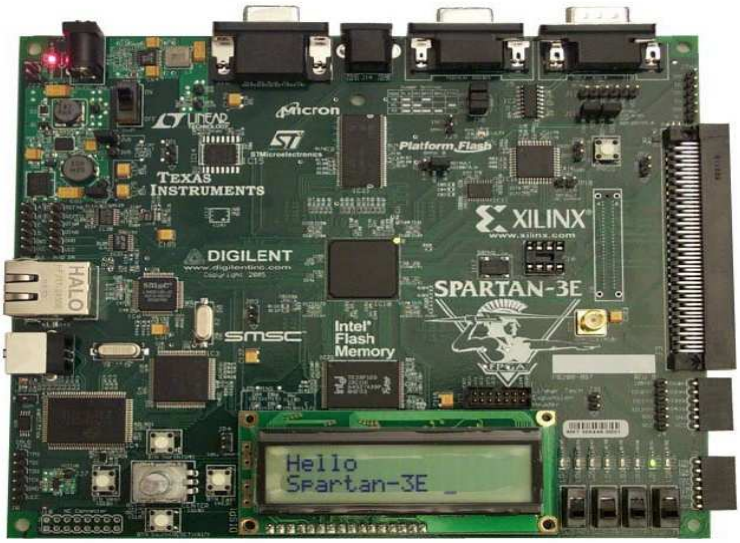



**Figura [6], Diseño plataforma FPGA**
**Fuente:** http://cephis.uab.es/resources/pdf/papers/JCRA_2004_Disseny.pdf

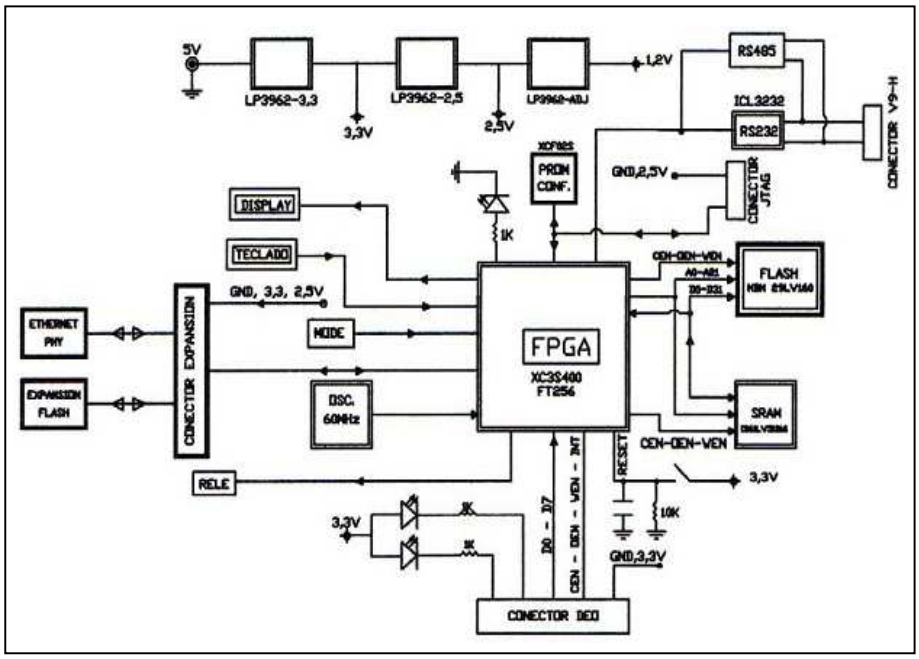



**Figura [7], Diagrama de bloques FPGA**
**Fuente:** http://www.microafis.com/paginas/spartan3.pdf

### 2.4.1 Puertos Seriales

Como se muestra en la figura [8], el FPGA Spartan 3E dos puertos RS-232: un conector DB9 DCE hembra y un conector de DTE macho.

El puerto de DCE se conecta directamente al conector del puerto serie disponible en la mayoría de los ordenadores personales y estaciones de trabajo a través de un cable estándar serie.

Utilice el conector de DTE para controlar otros RS-232 periféricos, como módems o impresoras, o realizar pruebas de bucle simple con el conector DCE.

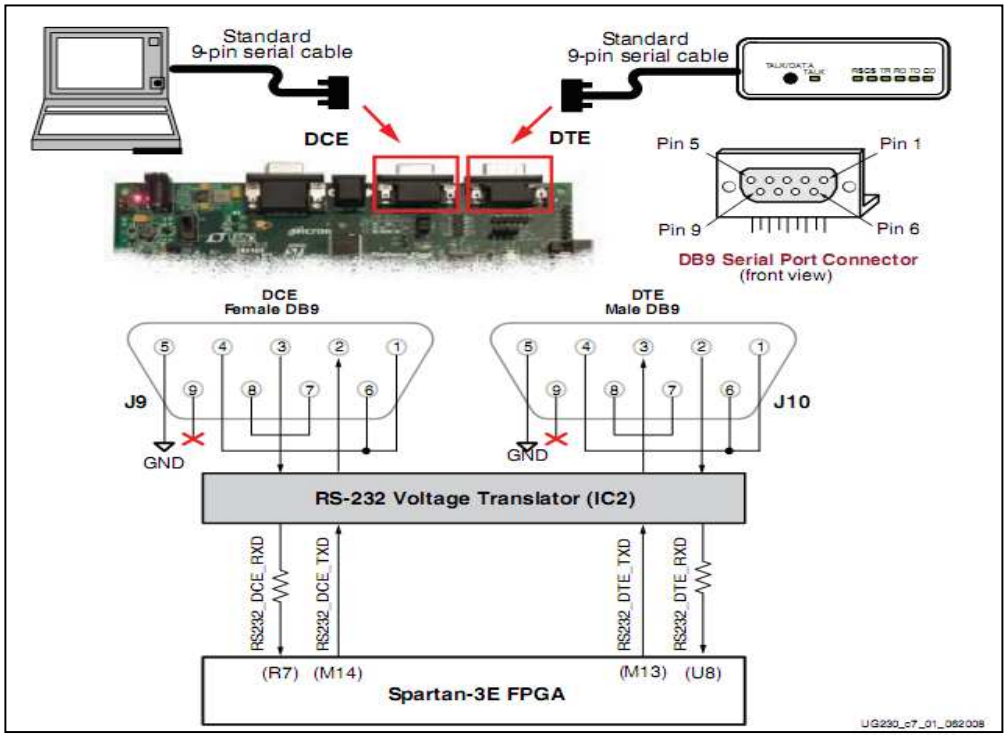



**Figura [8], Puertos seriales RS232**
**Fuente:** http://www.xilinx.com/support/documentation/7_series_white_papers.htm

## 2.5 IMPLEMENTACIÓN EN FPGA

Para implementar un diseño en un FPGA se utilizan lenguajes de descripción de hardware HDL[9], el más conocido es VHDL[10], sin embargo, no es la única opción, existen otras alternativas, no obstante el VHDL tiene varias ventajas que se describirán en el Capítulo III.

VHDL fue desarrollado como un lenguaje para el modelado de sistemas digitales. Proporciona una sintaxis amplia y flexible que permite el modelado estructural, en flujo de datos y de comportamiento de hardware.
VHDL es un estándar de la IEEE, lo que favoreció su adopción en la industria lo que se ve reflejado en las constantes mejoras en las herramientas. Debido a su estandarización,

[9] Hardware Description Languages
[10] Very Hard Description Language

un código en VHDL puede ser portado a diferentes herramientas y también, puede ser reutilizado en diferentes diseños.

Handel -C es un lenguaje de programación, diseñado para permitir la compilación de programas en hardware. No es un lenguaje de descripción de hardware como VHDL pero permite implementar arquitecturas para hardware en alto nivel.

Es posible diseñar sistemas secuenciales, sin embargo, se obtiene un mayor beneficio en desempeño del hardware con el uso del paralelismo. Debido a que Handel -C es se basa en un lenguaje ampliamente difundido, es más natural para un programador expresar un algoritmo en un lenguaje de alto nivel, sin importar lo que el compilador tenga que realizar para producir un diseño lógico a nivel de compuertas.

### 2.5.1 Interconexiones Programables

Su función es de interconectar las entradas y salidas de los CLBs entre sí y con el exterior. Se debe considerar qué tecnología utilizar para programar las conexiones entre las pistas, ésta puede variar de acuerdo al fabricante.

Las formas más comunes de programar los FPGAs, para interconectar las celdas lógicas, es mediante dos tipos de tecnologías, que son: SRAM[11] y antifusibles[12]. Con celdas SRAMs, la información de la interconexión es obtenida mediante un proceso de configuración, en el momento del encendido del circuito que contiene al FPGA, ya que al ser SRAM el contenido de estos bloques de memoria se pierde cuando se deja de suministrar energía. Estas celdas SRAMs con la información de la configuración requerida, controlan compuertas de paso o multiplexores que permiten interconectar a los CLBs. Esta tecnología generalmente es usada por dispositivos con celdas lógicas grandes.

[11] Static Random Access Memory (SRAM), o Memoria Estática de Acceso Aleatorio
[12] Se llama antifusible (antifuse) a un dispositivo programable

Antifusibles es una tecnología similar a la tecnología PROM[13], por lo que un FPGA sólo se puede programar una vez. Para la programación se utiliza algo similar a un fusible; la diferencia radica en que en los fusibles normales se desactivan deshabilitando la conexión, en tanto que en los antifusibles, cuando son programados, se produce una conexión, por lo que normalmente se encuentran abiertos. Los fusibles que no se requieren para la información que se desea almacenar, son quemados, por lo que no son reutilizables.

La tecnología SRAM es utilizada por fabricantes como: Altera, Lucent Technologies, Atmel, Xilinx y otros. La tecnología ANTIFUSE es utilizada por Cypress, Actel, QuickLogic, y Xilinx.

## 2.6 ASIGNACIÓN DE ENTRADAS Y SALIDAS

Estos bloques son los encargados de comunicar a todo el circuito integrado con el exterior. Cada IOB controla un pin del encapsulado; puede ser configurable como entrada, salida o bidireccional. En la siguiente Figura se muestra un esquema de un IOB. **(VER ANEXO 1)**.

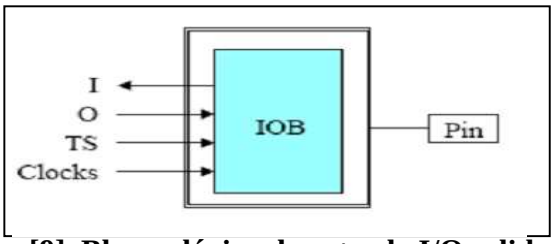



**Figura [9], Bloque lógico de entrada I/O salida (IOB)**

**Fuente:**

http://www.microcontroladorespic.com/tutoriales/FPGAs/estructura'configuracion.html

[13] Memoria ROM programable

Los IOBs se pueden implementar con flip-flops[14], latches[15], o circuitos combinacionales. En algunos casos la salida que presentan es triestado[16] (TS).

Al igual que en los CPLDs[17], su función es permitir el paso de la señal dentro y hacia fuera del dispositivo. Dependiendo del fabricante un IOB puede o no ser considerado como parte del bloque lógico.

## 2.7 LENGUAJE DE DESCRIPCIÓN DE HARDWARE

### 2.7.1 El lenguaje VHDL

La primera forma que apareció para describir un circuito, consistía en usar esquemas, es decir, mediante una descripción gráfica. Posteriormente, aparecieron herramientas para describir un circuito mediante sentencias o como un listado de conexiones, este tipo de lenguaje se denominó Netlist[18].

Con el gran impulso tecnológico que tomaron los circuitos integrados y dispositivos de lógica programable, se presentó la necesidad de describir circuitos en un alto nivel de abstracción, no como una lista de conexiones, sino desde el punto de vista funcional, fue entonces cuando VHDL se convirtió en un estándar.

VHDL fue desarrollado como un lenguaje para modelado y simulación lógica de circuitos electrónicos. El modelado permite describir a un circuito en distintos estilos de comportamiento y crear un modelo del mismo, cuyo comportamiento es conocido, para luego ser simulado. Los modelos de simulación son útiles para comprobar el

[14] Circuitos lógicos encargados de almacenar la información
[15] Circuito electrónico usado para almacenar información en sistemas lógicos asíncronos
[16] La lógica triestado permite puertos de salida con valor 0,1
[17] Un CPLD (del acrónimo inglés Complex Programmable Logic Device) es un dispositivo electrónico
[18] Describe la conectividad de un diseño electrónico.

correcto funcionamiento de un circuito. Posteriormente, su campo de aplicación fue expandiéndose a otras áreas que venían desarrollándose, como: síntesis, modelado de rendimiento, diagnóstico de fallos y documentación.

Las múltiples facilidades y ventajas que posee VHDL, con respecto a otros HDLs, constituyeron una motivación para escoger a este lenguaje como base de este proyecto, además que el hardware disponible (sistema UP2) posee compatibilidad con VHDL. Entre los beneficios que proporciona VHDL se mencionan los más importantes:

- Permite modelar, diseñar y simular desde un alto nivel de abstracción hasta un nivel más bajo; además, permite un diseño modular que consiste en dividir o descomponer un diseño en bloques o módulos más pequeños e independientes.
- Tiene capacidad descriptiva para múltiples modelos y niveles de abstracción (funcional, estructural y flujo de datos).
- Es un lenguaje independiente de la tecnología y de los fabricantes; es decir, no es ajustado a un determinado simulador y no requiere una metodología precisa de diseño.
- Permite implementar diseños existentes en nuevas tecnologías, es decir, tiene posibilidad de reutilización.
- Dispone de una gran versatilidad para la descripción de sistemas complejos, por lo que posee una sintaxis amplia y flexible.
- Las características antes mencionadas y otras adicionales de menor magnitud, han convertido a VHDL en un lenguaje de gran difusión.

### 2.7.2 Identificadores

Los identificadores son un conjunto de caracteres, con los cuales podemos representar diferentes elementos dentro de una descripción. Son etiquetas que identifican una variable, una señal, etc. Los nombres que se escojan para un identificador deben estar

dispuestos de una forma adecuada y siguiendo normas propias del lenguaje. Las reglas a tener en cuenta se presentan a continuación:

Los identificadores deben empezar con un carácter alfabético, no pudiendo terminar con un carácter subrayado, ni tener dos o más de estos caracteres subrayados seguidos.

VHDL permite utilización de letras mayúsculas (A…Z) minúsculas (a…z) dígitos (0…9) y el carácter subrayado (_). No hay distinción entre mayúsculas y minúsculas.

No puede usarse como identificador una palabra reservada por VHDL.

Ejemplos de identificadores:

**Mux4a2 Mi_entidad clock_10khz**

**2.7.2.1 Objetos de Datos**

Un objeto en VHDL es un elemento que tiene asignado un valor, de un tipo de datos determinado. Según sea el tipo de dato, el objeto poseerá un conjunto de operaciones que se le podrán aplicar. En general, no será posible realizar operaciones entre dos objetos de distinto tipo, a menos que se defina previamente un programa de conversión de tipos. Usualmente, los objetos de datos son:

**2.7.2.1.1 Constantes:** Son elementos que pueden tomar un único valor de un tipo dado. Si la declaración está dentro de una arquitectura, entidad, proceso, paquete u otra estructura del lenguaje, solo pueden ser utilizadas dentro de la estructura correspondiente.

**2.7.2.1.2 Variables:** Son elementos cuyo valor puede ser modificado cuando sea necesario. Se les puede asignar un valor inicial al momento de ser declaradas. Las variables se utilizan únicamente dentro de procesos y subprogramas; se usan

generalmente como índices de bucles, o para tomar valores y modelar componentes; no se usan para representar conexiones o estados de memoria.

**2.7.2.1.3 Señales:** Son similares a las variables, con la diferencia que las señales si pueden tomar valores lógicos, mientras que las variables no lo pueden hacer, por lo que si pueden representar conexiones o elementos de memoria.

**2.7.2.1.4 Alias:** No es un objeto de datos en sí, son tramos o segmentos de un objeto de datos ya existente.

En la siguiente tabla se muestra los objetos de datos dentro de VHDL.

| **Objetos de Datos** | **Síntesis de Declaración** | **Ejemplo** |
|---|---|---|
| **Constantes** | CONSTANT identificador: tipo: = valor | CONSTANT byte: integer: =5; |
| **Variables** | VARIABLE identificador: tipo [:=valor inicial]; | VARIABLE aux1, aux2, bit; |
| **Señales** | SIGNAL identificador: tipo [:=valor inicial]; | SIGNAL x,y: bit := 0;<br>SIGNAL dato: bit_vector (7 DOWNTO 0); |
| **Alias** | ALIAS **identif**: tipo IS identif2 rango | ALIAS instr: bit_vector (3 DOWNTO 0) IS dato (7 DOWNTO 4) |

**Tabla [1] Declaración de objetos de datos**
**Fuente:** http://www.jimenez-ruiz.es/ernesto/II/VHDL/vhdl.html

### 2.7.2.2 Diferencia entre señales y variables

Es preciso mencionar que las variables y señales, son susceptibles a posibles confusiones, es por eso que se exponen claramente las diferencias entre ellas.

La diferencia substancial radica en que las señales pueden ser empleadas en elementos lógicos y/o conexiones; representan un nodo de conexión entre elementos lógicos (compuertas, registros, buffers, etc.); además, en la sintaxis, el símbolo de asignación es diferente, para las señales se utiliza como símbolo: "<=" y para las variables el símbolo: ":="; únicamente cuando se usa un valor inicial (en la declaración), en ambos casos se utiliza el símbolo: ":=".

En la siguiente tabla se muestra la diferencia entre señales y variables.

| **Objeto de Datos** | **Síntesis para Asignación** | **Ejemplos** |
|---|---|---|
| **Variables** | Identificador: = Expresión; | V:=V+1;<br>(V,W):=X; |
| **Señales** | Identificador <= Expresión<br>Identificador <= [Options]<br>Expresión [after time ns],<br>Expresión [after time ns],<br>--La palabra after es opcional y<br>significa retraso de tiempo | A<=B;<br>A<=B nand C;<br>A<=B nand C after 0.2 ns;<br><br>H<=”00”,<br>“01” after 10 ns,<br>“10” after 20 ns; |

**Tabla [2], Asignación de objetos de datos**
**Fuente:** http://www.jimenez-ruiz.es/ernesto/II/VHDL/vhdl.html

Las variables son análogas a las usadas en cualquier otro lenguaje de programación; éstas toman un valor exactamente en el instante de su asignación y sólo pueden ser declaradas dentro de procesos (véase 2.2.2). Las señales dentro de un proceso tienen un valor inicial y un valor final; dentro del proceso el valor inicial asignado puede ser modificado y sólo cuando el proceso haya culminado, toman el valor final. Las señales son declaradas en entidades o arquitecturas; las señales que son declaradas en las entidades, corresponden a entradas y/o salidas externas al circuito o módulo (Unión de dos o más circuitos) y las señales declaradas en la arquitectura corresponden a interconexiones internas.

En la Figura [10] se muestra un ejemplo; las señales externas son: X1,X2,X3,X4,Y, y las señales internas corresponden a T1,T2,T3,T4,U1,U2,V1,W1. Como la señal representa un nodo de conexión o unión de cables, ésta puede tomar uno o diferentes nombres (U1 ó U2; W1 ó Y), sin modificar la estructura general del circuito.

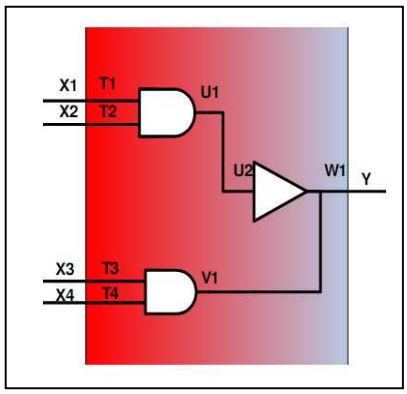



**Figura [10], Señales internas y externas de un circuito o módulo**
**Fuente:** http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/928/4/T10691CAP1.pdf

### 2.7.2.3 Tipos de Datos

Los valores que pueden tomar y las operaciones que pueden realizar los objetos de datos, son limitados por el tipo de datos que posee cada objeto. Aparte de los tipos predefinidos, se pueden crear nuevos tipos, e incluso subtipos; que básicamente son

subconjuntos de tipos, ya sean de tipos predefinidos o de los nuevos tipos creados. Los tipos usados para crear los subtipos usualmente se conocen como tipos base.

No todas las operaciones se pueden utilizar con los diferentes tipos de datos a menos que se utilicen librerías adecuadas, en las que estén definidas funciones para la conversión de tipos.

Los tipos de datos se clasifican de acuerdo a la función que van a desempeñar los objetos de datos dentro de una arquitectura o un proceso. La Figura 9 muestra dicha clasificación, donde los tipos de datos que están sombreados, no son muy utilizados, ni tampoco soportados en VHDL para síntesis, debido a que no tienen ningún objeto o función para describir hardware; estos tipos de datos más bien son utilizados en simulación. El presente proyecto se fundamenta en la síntesis, por lo cual no se usará este tipo de datos y por ende no se entrará en detalles acerca de los mismos.

Para crear nuevos tipos se utiliza la palabra reservada type, y para crear subtipos la palabra subtype. La sintaxis que se utiliza, se puede apreciar en los ejemplos de los diferentes tipos de datos, que se presentan más adelante.

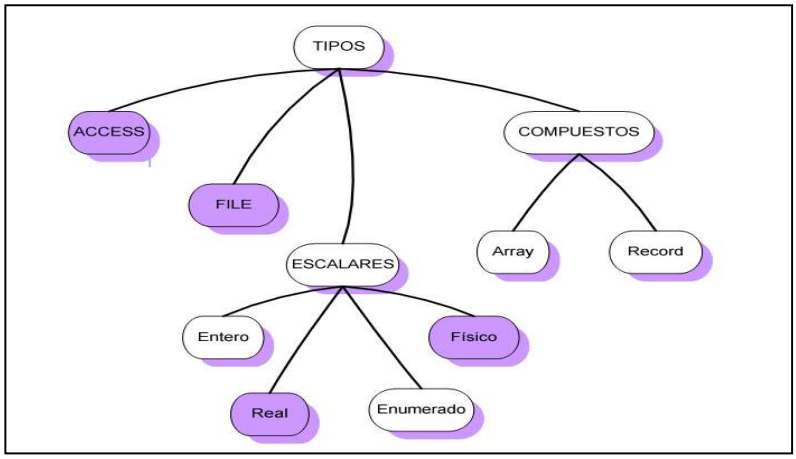



**Figura [11], Tipos de datos**
**Fuente:** http://www.fic.udc.es/files/asignaturas/83TS/2.TiposDeDatos.pdf

Una vez creado el tipo de datos, éste es especificado cuando se declara objetos de datos; en caso de utilizar los tipos de datos predefinidos por el lenguaje, basta con mencionar el nombre del tipo de dato en la declaración, pues estos tipos de datos ya fueron creados.

En el ejemplo siguiente se crea un tipo de datos llamado: “ESTADOS“, el mismo que es usado para declarar objetos de este tipo.

```
TYPE ESTADOS IS (ESTADO_A, ESTADO_B, ESTADO_C);
VARIABLE EST: ESTADOS;
```

### 2.7.3 Estructura de un programa en VHDL

A continuación se muestra un ejemplo Transmisor / Receptor serie asíncrono conforme a la norma RS232.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Elaborado por Richard Osorio – Darwin Castillo
---- any Xilinx primitives in this code.


entity voting_circuit_2_of_3 is
    Port ( clk : in std_logic;
           reset : in std_logic;
           load_sample_1 : in std_logic;
           load_sample_2 : in std_logic;
           load_sample_3 : in std_logic;
           bit_input : in std_logic;
```

```
        sampled_bit : out std_logic;
        discrepancy : out std_logic);
end voting_circuit_2_of_3;

architecture Behavioral of voting_circuit_2_of_3 is

-- Signals declaration
signal sample_1, sample_2, sample_3: std_logic;
signal sample_vector: std_logic_vector (2 downto 0);

begin

-- Vector of samples
sample_vector <= sample_3 & sample_2 & sample_1;

-- Sample 1 register
Sample_1_register: process (clk, reset, load_sample_1)
begin
if reset = '1' then
                                sample_1 <= '0';
elsif clk'event and clk ='1' then
                                if load_sample_1 = '1' then sample_1 <= bit_input;
                                end if;
end if;
end process;

-- Sample 2 register
Sample_2_register: process (clk, reset, load_sample_2)
begin
if reset = '1' then
                                sample_2 <= '0';
```

```
elsif clk'event and clk ='1' then
                    if load_sample_2 = '1' then sample_2 <= bit_input;
                    end if;
end if;
end process;

-- Sample 3 register
Sample_3_register: process (clk, reset, load_sample_3)
begin
if reset = '1' then
                    sample_3 <= '0';
elsif clk'event and clk ='1' then
                    if load_sample_3 = '1' then sample_3 <= bit_input;
                    end if;
end if;
end process;

-- Voting circuit (2 of 3)
with sample_vector select
                    sampled_bit <= '1' when "011"|"101"|"110"|"111",
                                        '0' when others;

with sample_vector select
                    discrepancy <= '0' when "000"|"111",
                                        '1' when others;

end Behavioral;
```

En el ejemplo, anterior se muestra como realizar transmisión / recepción dentro del Lenguaje VHDL, y este se puede adaptar a tecnología inalámbrica, utilizando dispositivos con tecnología Zigbee.

## 2.8 TECNOLOGÍA ZIGBEE

### 2.8.1 Definiciones

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (wireless personal área network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

En principio, el ámbito donde se prevé que esta tecnología cobre más fuerza es en domótica, como puede verse en los documentos de la ZigBee Alliance, en las referencias bibliográficas que se dan más abajo es el documento «ZigBee y Domótica». La razón de ello son diversas características que lo diferencian de otras tecnologías:

- Su bajo consumo
- Su topología de red en malla
- Su fácil integración (se pueden fabricar nodos con muy poca electrónica).

### 2.8.2 Usos

Los protocolos ZigBee están definidos para su uso en aplicaciones encastadas con requerimientos muy bajos de transmisión de datos y consumo energético.

Se pretende su uso en aplicaciones de propósito general con características auto organizativas y bajo costo (redes en malla, en concreto). Puede utilizarse para realizar control industrial, albergar sensores empotrados, recolectar datos médicos, ejercer labores de detección de humo o intrusos o domótica. La red en su conjunto utilizará una cantidad muy pequeña de energía de forma que cada dispositivo individual pueda tener

una autonomía de hasta 5 años antes de necesitar un recambio en su sistema de alimentación.

### 2.8.3 Funcionalidad

Basándose en su funcionalidad, puede plantearse una segunda clasificación:
Dispositivo de funcionalidad completa (FFD): También conocidos como nodo activo. Es capaz de recibir mensajes en formato 802.15.4. Gracias a la memoria adicional y a la capacidad de computar, puede funcionar como Coordinador o Router ZigBee, o puede ser usado en dispositivos de red que actúen de interface con los usuarios.

Dispositivo de funcionalidad reducida (RFD): También conocido como nodo pasivo. Tiene capacidad y funcionalidad limitadas (especificada en el estándar) con el objetivo de conseguir un bajo coste y una gran simplicidad. Básicamente, son los sensores/actuadores de la red.

Un nodo ZigBee (tanto activo como pasivo) reduce su consumo gracias a que puede permanecer dormido la mayor parte del tiempo (incluso muchos días seguidos). Cuando se requiere su uso, el nodo ZigBee es capaz de despertar en un tiempo ínfimo, para volverse a dormir cuando deje de ser requerido. Un nodo cualquiera despierta en aproximadamente 15 ms.

### 2.8.4 Topologías

ZigBee permite tres topologías de red:

- Topología en estrella: el coordinador se sitúa en el centro.
- Topología en árbol: el coordinador será la raíz del árbol.
- Topología de malla: al menos uno de los nodos tendrá más de dos conexiones.

La topología más interesante (y una de las causas por las que parece que puede triunfar ZigBee) es la topología de malla. Ésta permite que si, en un momento dado, un nodo del camino falla y se cae, pueda seguir la comunicación entre todos los demás nodos debido a que se rehacen todos los caminos. La gestión de los caminos es tarea del coordinador.

Estrategias de conexión de los dispositivos en una red Zigbee

Las redes ZigBee han sido diseñadas para conservar la potencia en los nodos ‘esclavos’. De esta forma se consigue el bajo consumo de potencia. La estrategia consiste en que, durante mucho tiempo, un dispositivo "esclavo" está en modo "dormido", de tal forma que solo se "despierta" por una fracción de segundo para confirmar que está "vivo" en la red de dispositivos de la que forma parte. Esta transición del modo "dormido" al modo "despierto" (modo en el que realmente transmite), dura unos 15ms, y la enumeración de "esclavos" dura alrededor de 30ms, como ya se ha comentado anteriormente.

En la siguiente figura, se muestra el diagrama de flujo del envió de un mensaje en una red Zigbee.

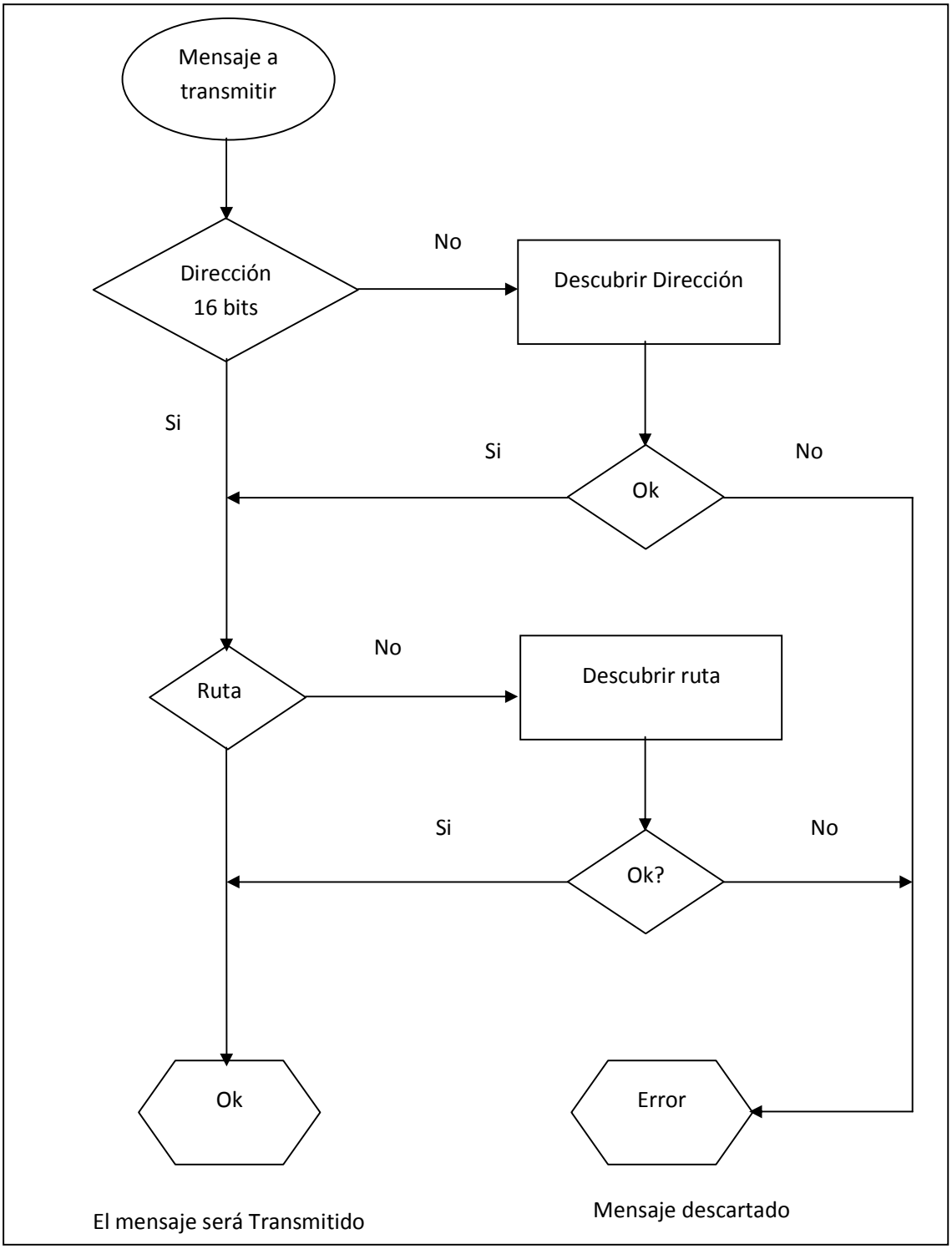



**Figura [12], Diagrama de Flujo Transmisión mediante Zigbee**
**Elaborado:** ***Autores de la Tesis***

### 2.8.5 Módulos Zigbee

Son módulos de radio frecuencia diseñados para operar bajo el protocolo Zigbee, cuando se requiere una comunicación segura entre dispositivos remotos con un bajo consumo de energía, son utilizados en su mayoría en aplicaciones de automatización de casas (domótica), sistema de seguridad, monitoreo de sistemas remotos, aparatos domésticos, etc.

Estos módulos operan dentro de la banda ISM[19] utilizando la frecuencia 2.4 GHz

Los módulos Zigbee tienes dos módulos de comunicación: Transmisión Serial Transparente (modo AT) y el modo API[20] los módulos Zigbee pueden ser configurados desde cualquier computador utilizando el software X-CTU o también desde el propio microcontrolador.

Existen dos series dentro de los módulos Xbee que son la serie 1 y la serie 2 o también llamada 2.5, los módulos serie 1 y serie 2 tienen el mismo pinout[21], sin embargo no son compatibles entre ellos porque utilizan distintos chipset y trabajan con protocolos diferentes.

Los módulos Zigbee serie 1 estan basados en el chipset[22] de freescale[23] para ser utilizado en redes punto y punto a multipunto, mientras que los módulos de la serie 2 están

[19] (Industrial Scientific and Medical) bandas reservadas internacionalmente para uso no comercial de radio frecuencia electromagnética en áreas industriales, científicas y medicas.

[20] Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

[21] Se puede describir cómo un conector es cableado. Cada patica del conector tiene un propósito que se describe brevemente en el pinout.

[22] Circuito integrado auxiliar o chipset es el conjunto de circuitos integrados diseñados con base a la arquitectura de un procesador (en algunos casos diseñados como parte integral de esa arquitectura), permitiendo que ese tipo de procesadores funcionen en una placa base. Sirven de puente de comunicación con el resto de componentes de la placa, como son la memoria, las tarjetas de expansión, los puertos USB, ratón, teclado, etc.

basados el chipset Ember[24] diseñados para ser utilizados en aplicaciones que requieren repetidores o una red Mesh. Tanto los módulos serie 1 y serie 2 pueden utilizar los modos AT y API.

### 2.8.5.1 Tipos de módulos Zigbee

En la siguiente, figura se muestra los distintos tipos de módulos Zigbee.

#### 2.8.5.1.1 Modulos Zigbee

Los modulos Zigbee tienen un alcance en interiores de hasta 30 metros y en exteriores de hasta 100 metros con antena dipolo que se muestra en la figura.

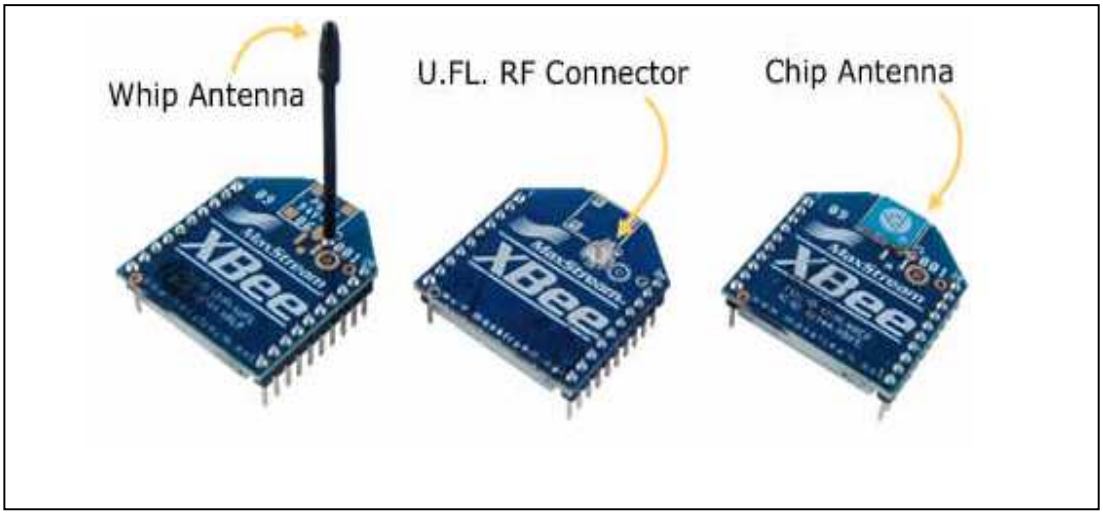



**Figura [13], Tipos de módulos Zigbee**
**Fuente:** http://decelectronics.com/html/XBEE/XBEE.htm

[23] Freescale Semiconductor, Inc. es un fabricante estadounidense de semiconductores. Fue creado a partir de la división de semiconductores de Motorola en 2004. Freescale se centra en el mercado de los sistemas integrados y las comunicaciones.
[24] Empresa líder proveedora de sensores inalámbricos y tecnologías de control de red.

#### 2.8.5.1.2 Requerimientos de Conexión

Para la conexión de los módulos Zigbee se requieren como mínimo proveer de dos terminales (VCC y GND) y dos de datos (Vin y Vout), para realizar actualizaciones de firmware en los modulos se tiene que agregar las conexiones de los pines RTS y DTR.

Los módulos Zigbee no requieren ser soldados porque su diseño les permite ser montados y desmontados de un zócalo que al igual que los módulos deben ser de dos hileras de 10 pines separadas entre ellas por 22 mm y una separación entre pines de 2 mm.

Los módulos Zigbee pueden ser programados a través de un Hyperterminal y una interface serial con un MAX232[25] y una serie de comandos AT convirtiéndose en un método muy complicado y tedioso por este motivo existen dos clases de interfaz serial y USB que con la ayuda del software X-CTU son utilizadas para definir parámetros del módulo Zigbee de una manera más rápida.

#### 2.8.5.1.3 Seguridad en los módulos Zigbee

Proveen un cifrado AES[26] de 128 bits esta clave es asignada manualmente y no es posible leerla, solo ingresarla, Además de cifrar la clave cifra todo lo que se transmite y setea la bandera correspondiente a las tramas de esta manera se transmite tramas seguras. La clave debe ser la misma para todos los dispositivos de la red que intervengan en la comunicación.

Los parámetros que manejan la seguridad son EE (Encription Enable) y la clave es almacenada en el campo KY.

---

[25] Circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL (tiempo de vida y viceversa).

[26] También conocido como Rijndael (pronunciado "Rain Doll" en inglés), es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos.

Los módulos Zigbee implementan seguridad de la siguiente manera:

- No poseen un manejo de claves dentro de las aplicaciones
- Poseen una clave de enlace única que se define previamente dentro del parámetro KY
- La clave de red única es enviada a todos los componentes de la red desde su coordinador esta clave no puede contener el valor de 0 y si fuera el caso se asignara aleatoriamente.
- El coordinador está en la capacidad del configurar el parámetro de seguridad EO (Encryption Option) con el bit 1 haciendo que la clave de red se actualice periódicamente.

### 2.8.6 Dispositivos Electrónicos

#### 2.8.6.1 Microcontroladores

Un microcontrolador es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida.

Son diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control de un electrodoméstico sencillo como una batidora, utilizará un procesador muy pequeño (4 u 8 bit) por que sustituirá a un autómata finito. En cambio un reproductor de música y/o vídeo digital (mp3 o mp4) requerirá de un procesador de 32 bit o de 64 bit y de uno o más Códec de señal digital (audio y/o vídeo). El control de un sistema de frenos ABS (Antilock Brake System) se basa normalmente en un microcontrolador de 16 bit, al igual que el sistema de control electrónico del motor en un automóvil.

Los microcontroladores representan la inmensa mayoría de los chips de computadoras vendidos, sobre un 50% son controladores "simples" y el restante corresponde a DSPs más especializados. Mientras se pueden tener uno o dos microprocesadores de propósito general en casa (Ud. está usando uno para esto), usted tiene distribuidos seguramente entre los electrodomésticos de su hogar una o dos docenas de microcontroladores. Pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos microondas, teléfonos, etc.

#### 2.8.6.1.1 Componentes del microcontrolador

##### 2.8.6.1.1.1 El UDC o procesador

El procesador (CPU, por Central Processing Unit o Unidad Central de Procesamiento), es por decirlo de alguna manera, el cerebro del ordenador. Permite el procesamiento de información numérica, es decir, información ingresada en formato binario, así como la ejecución de instrucciones almacenadas en la memoria.

El primer microprocesador (Intel 4004) se inventó en 1971. Era un dispositivo de cálculo de 4 bits, con una velocidad de 108 kHz. Desde entonces, la potencia de los microprocesadores ha aumentado de manera exponencial.

##### 2.8.6.1.1.2 Memoria

En los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos.

Hay dos peculiaridades que diferencian a los microcontroladores de los computadores personales:

- No existen sistemas de almacenamiento masivo como disco duro o disquetes.

- Como el microcontrolador sólo se destina a una tarea en la memoria ROM, sólo hay que almacenar un único programa de trabajo.

La RAM en estos dispositivos es de poca capacidad pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM pues se ejecuta directamente desde la ROM.

Los usuarios de computadores personales están habituados a manejar Megabytes de memoria, pero, los diseñadores con microcontroladores trabajan con capacidades de ROM comprendidas entre 512 bytes y 8 k bytes y de RAM comprendidas entre 20 y 512 bytes.

Según el tipo de memoria ROM que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Se describen las cinco versiones de memoria no volátil que se pueden encontrar en los microcontroladores del mercado.

#### 2.8.6.1.1.2.1 ROM con máscara

Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip. El elevado coste del diseño de la máscara sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

#### 2.8.6.1.1.2.2 OTP

El microcontrolador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario. OTP (One Time Programmable). Es el usuario quien puede

escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC.

La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas.

Tanto en este tipo de memoria como en la EPROM, se suele usar la encriptación mediante fusibles para proteger el código contenido.

#### 2.8.6.1.1.2.3 EPROM

Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read OnIy Memory) pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.

#### 2.8.6.1.1.2.4 EEPROM

Se trata de memorias de sólo lectura, programables y borrables eléctricamente EEPROM (Electrical Erasable Programmable Read OnIy Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y la de borrado. No disponen de ventana de cristal en la superficie.

Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

El número de veces que puede grabarse y borrarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación contínua. Son muy idóneos para la enseñanza y la Ingeniería de diseño.

Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecuan el dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

**2.8.6.1.1.2.5 FLASH**

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña.

A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM.

La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado.

Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados "en circuito", es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado al control del motor de un automóvil permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores tales como la compresión, la instalación de nuevas piezas, etc. La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

#### 2.8.6.1.1.2.6 Puertas de Entrada y Salida

La principal utilidad de las patitas que posee la cápsula que contiene un microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores.

Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

#### 2.8.6.1.1.3 Reloj principal

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito de reloj está incorporado en el microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía.

### 2.8.7 RECURSOS ESPECIALES

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador. En algunas amplía las capacidades de las memorias, en otras incorpora nuevos recursos, en otras reduce las prestaciones al mínimo para aplicaciones muy simples, etc. La labor del diseñador es encontrar el modelo mínimo que satisfaga todos los requerimientos de su aplicación. De esta forma, minimizará el coste, el hardware y el software.

Los principales recursos específicos que incorporan los microcontroladores son:

• Temporizadores o "Timers".
• Perro guardián o "Watchdog".
• Protección ante fallo de alimentación o "Brownout".
• Estado de reposo o de bajo consumo.
• Conversor A/D.
• Conversor D/A.
• Comparador analógico.
• Modulador de anchura de impulsos o PWM.
• Puertas de E/S digitales.
• Puertas de comunicación.

#### 2.8.7.1 Temporizadores o "Timers"

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

#### 2.8.7.2 Perro guardián o "Watchdog"

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, "ladrará y ladrará" hasta provocar el reset.

#### 2.8.7.3 Protección ante fallo de alimentación o "Brownout"

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo ("brownout"). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

#### 2.8.7.4 Estado de reposo ó de bajo consumo

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (SLEEP en los PIC), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando sumido en un profundo "sueño" el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

#### 2.8.7.4.1 Conversor A/D (CAD)

Los microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patitas del circuito integrado.

#### 2.8.7.4.2 Conversor D/A (CDA)

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patitas de la cápsula. Existen muchos efectores que trabajan con señales analógicas.

#### 2.8.7.4.3 Comparador analógico

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las patitas de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.
También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

#### 2.8.7.4.4 Modulador de anchura de impulsos o PWM

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

#### 2.8.7.4.5 Puertos de E/S digitales

Todos los microcontroladores destinan algunas de sus patitas a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos.

Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

#### 2.8.7.4.6 Puertos de comunicación

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- UART, adaptador de comunicación serie asíncrona.
- USART, adaptador de comunicación serie síncrona y asíncrona
- Puerta paralela esclava para poder conectarse con los buses de otros microprocesadores.
- USB (Universal Serial Bus), que es un moderno bus serie para los PC.
- Bus I2C, que es un interfaz serie de dos hilos desarrollado por Philips.
- CAN (Controller Área Network), para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU. se usa el J185O.

### 2.8.8 Capas de una red Zigbee

En la arquitectura Zigbee se tiene las siguientes capas:

#### 2.8.8.1 Capa Física

Trabaja en cualquiera de estos rangos de frecuencia: 868 MHz, 915 MHz, o 2.4 GHz, alcanzando velocidades de 20 Kbps, 40 Kbps y 250 Kbps, permitiendo alcanzar distancias de una máximo de 100 metros con un muy bajo consumo de energía haciéndole ideal para aplicaciones domóticas.

#### 2.8.8.2 Capa MAC

Proporciona servicios que permiten la fiabilidad y la comunicación directa entre dispositivos.

Zigbee aporta con más capas al estándar IEEE 802.15.4 las cuales son:

#### 2.8.8.3 Capa de red NWK (Network Layer)

Diseñada para gestionar el enrutamiento y mantenimiento de los demás nodos que son parte de la red.

#### 2.8.8.4 Capa Aplicación (APL Application Layer)

Se subdivide en la subcapa APS (Application Support Layer) que tiene como función dar una interfaz a la capa red y a la subcapa ZDO (Zigbee Device Object) encargada de inicializar la subcapa APS y la subcapa NWK.

De esta manera Zigbee conjuntamente con IEEE 802.15.4 conforman una arquitectura completa de protocolos la cual permite la comunicación de una gran cantidad de dispositivos o nodos dentro de una misma red.

Todas las transmisiones que se realizan mediante Zigbee en la banda 2.4 GHz especifican 16 canales (11 al 26) de los cuales solo se ocupara uno, el cual es capaz de tener actividad dentro de otras tecnologías. Para escoger el canal apropiado para trabajar se evalua el nivel de la señal mediante 2 procedimientos.

ED (Energy Detection) y CCA (Clear Channel Assessment) los que evalúan una señal compatible con 802.15.4 y el nivel de energía presente en el canal. Zigbee al estar en una banda libre permite la coexistencia de otras bandas similares como 802.11 b – g y proveen alternativas para sitios donde existen una proliferación de redes WIFI.

Teniendo el antecedente de las capas de una red Zigbee es necesario describir los sensores y actuadores que se utiliza, ya que este es el medio que realizará la transmisión, para este prototipo se utilizará: Sensor fotovoltaico, detector de movimiento, detector de humo.

## 2.9 SENSORES Y ACTUADORES

### 2.9.1 Sensor

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica (como en una RTD), una capacidad eléctrica (como en un sensor de humedad), una Tensión eléctrica (como en un termopar), una corriente eléctrica (como en un fototransistor), etc.

### 2.9.2 Actuador

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control como, por ejemplo, una válvula.

Existen varios tipos de actuadores como son:

- Electrónicos
- Hidráulicos
- Neumáticos
- Eléctricos

Los actuadores hidráulicos, neumáticos y eléctricos son usados para manejar aparatos mecatrónicos. Por lo general, los actuadores hidráulicos se emplean cuando lo que se necesita es potencia, y los neumáticos son simples posicionamientos. Sin embargo, los hidráulicos requieren mucho equipo para suministro de energía, así como de mantenimiento periódico. Por otro lado, las aplicaciones de los modelos neumáticos también son limitadas desde el punto de vista de precisión y mantenimiento.

### 2.9.3 Sensor Fotovoltaico

Este sensor es capaz de detectar o medir el nivel de luz en un espacio físico determinado y proporcionar una señal de salida dependiendo de la intensidad de la luz a la que está expuesta.

El funcionamiento básico de estos sensores se basa en convertir la luz en una señal eléctrica.

### 2.9.4 Detector de movimiento

Un detector de movimiento es un dispositivo electrónico equipado de sensores que responden un movimiento físico. Se encuentran, generalmente, en sistemas de seguridad o en circuitos cerrados de televisión. El sistema puede estar compuesto, simplemente, por una cámara de vigilancia conectada a un ordenador, que se encarga de generar una señal de alarma o poner el sistema en estado de alerta cuando algo se mueve delante de la cámara.

Aunque, para mejorar el sistema se suele utilizar más de una cámara, multiplexores y grabadores digitales.

### 2.9.5 Detector de Humo

Un detector de humo es un aparato de seguridad que detecta la presencia de humo en el aire y emite una señal acústica avisando del peligro de incendio. Atendiendo al método de detección que usan, pueden ser de dos tipos: ópticos o iónicos, aunque algunos usen los dos mecanismos para aumentar su eficacia.

## 2.10 REDES DE ÁREA PERSONAL PAN

### 2.10.1 Origen.

PAN es una solución de red que aumenta nuestro ambiente personal de trabajo, seguridad. Se tiene una gran variedad de dispositivos disponibles (PDAs, organizadores personales Webpads, computadores de mano, cámaras, etc). Que envuelven a una persona dentro de la distancia que puede ser cubierta por la voz y proveen comunicaciones captables dentro del espacio personal y con el mundo exterior. Para los próximos dispositivos que trabajan en estas redes, PAN puede usar el medio inalámbrico, el campo eléctrico del cuerpo humano como un conductor, así como el

campo magnético. En particular cuando se usa el medio inalámbrico, Se refiere a las PANs inalámbricas. Nos referimos a la tecnología WPAN.

La WPAN forma una burbuja alrededor de la persona, que se le denomina ESPACIO DE OPERACIÓN PERSONAL (POS). Un concepto fundamental anterior a los sistemas WPAN, afirma que en algún momento dos dispositivos equipados con WPAN tienen cobertura dentro de aproximadamente 10 metros uno del otro (sus POSs se cruzan, intersecan). Se forma entonces una posible conexión. Un dispositivo WPAN puede conectarse a un repetidor para acceder al internet ó puede ser dinámicamente extendido para incluir el acceso a sensores y actuadores.

La tecnología inalámbrica WPAN es de corto-alance, en relación a la tecnología WLAN. Sin embargo WLAN y WPAN tienen situaciones complementarias:

WPAN enfatiza el bajo costo y el bajo consumo de potencia, usualmente permite una rápida transmisión y un máximo flujo de datos. WLAN enfatiza sobre una gran cantidad de datos y un amplio rango de cobertura que representa un gasto en los costos y el consumo de potencia.

**2.10.2 Protocolo**

Los protocolos MAC tienen la tarea fundamental de evitar colisiones, para que dos nodos interferentes no transmitan al mismo tiempo. Las colisiones son una causa de desperdicio de energía, pero no es la única. Otras fuentes de desperdicio de energía son:

Paquete de control Overhead: Consume energía al enviar y recibir paquetes;
Overhearing: Un nodo recoge paquetes dirigidos a otros nodos;
Desocupado escuchando [idle listening]: Escucha para recibir los posibles paquetes que no han llegado.

Muchas medidas hechas por IEEE 802.11 han mostrado que idle listening consume de 50 a 100% de la energía requerida por el receptor. Un protocolo MAC de potencia eficiente debería reducir el gasto de energía de todas las fuentes mencionadas.

En consecuencia, la necesidad de soportar diferentes clases de servicios con varias garantías QoS típicamente requieren varios tipos de eventos. Esto está en contraste con la naturaleza aleatoria de la mayoría de protocolos MAC actual. Es requerido un delicado trade-off entre usuarios previstos [sheduled users] y usuarios de mejor esfuerzo [best efford users]. El desarrollo de protocolos de acceso aleatorio con varios tipos de servicios de diferenciación todavía propone muchos problemas abiertos.

El mayor ahorro de energía viene de la disponibilidad de hardware de disminuir potencia en elementos de sistemas seleccionados que no la requieren. IEEE 802.11 tiene un modo de ahorro de energía donde un nodo solo necesita estar despierto periódicamente.

Otro aspecto de manejo de energía es para incrementar el tiempo de vida de la batería de un nodo móvil usando técnicas de manejo de baterías de energía eficiente como una política adecuada de descarga de batería.

Una batería consiste de varias celdas electroquímicas en las que la potencia necesita ser drenada cuando el nodo transmite un paquete. Cuando en una celda es permitido descansar entre periodos de descarga, ésta es capaz de recuperar parte de esa carga, gracias a los mecanismos de difusión. Una política de descarga de batería decide que celdas [celdas electroquímicas] debería servir al paquete y en qué celdas está permitido descansar. Resultados analíticos y de simulación han mostrado que las políticas de descarga tienen impacto significante en el tiempo de vida de la batería.

# CAPITULO III

## 3. ANALISIS Y REQUERIMIENTOS

En este capítulo, se realiza un análisis previo al diseño del prototipo, el cual dará a conocer los requerimientos necesarios y adecuados y mediante el estudio de estos, se escogerá cual es el más apropiado y que se ajuste al prototipo.

### 3.1 Análisis de tecnologías inalámbricas

#### 3.1.1 La radio frecuencia

El término radiofrecuencia, también denominado espectro de radiofrecuencia o RF, se aplica a la porción menos energética del espectro electromagnético, situada entre unos 3 Hz y unos 300 GHz. El Hertz es la unidad de medida de la frecuencia de las ondas, y corresponde a un ciclo por segundo.1 Las ondas electromagnéticas de esta región del espectro se pueden transmitir aplicando la corriente alterna originada en un generador a una antena.

A partir de 1 GHz las bandas entran dentro del espectro de las microondas. Por encima de 300 GHz la absorción de la radiación electromagnética por la atmósfera terrestre es tan alta que la atmósfera se vuelve opaca a ella, hasta que, en los denominados rangos de frecuencia infrarrojos y ópticos, vuelve de nuevo a ser transparente.

Las bandas ELF, SLF, ULF y VLF comparten el espectro de la AF (audiofrecuencia), que se encuentra entre 20 y 20.000 Hz aproximadamente. Sin embargo, éstas se tratan de ondas de presión, como el sonido, por lo que se desplazan a la velocidad del sonido sobre un medio material. Mientras que las ondas de radiofrecuencia, al ser ondas electromagnéticas, se desplazan a la velocidad de la luz y sin necesidad de un medio material.

#### 3.1.1.1 Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal, como PDA, teléfonos móviles, computadoras portátiles, computadoras personales, impresoras o cámaras digitales.

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basada en transceptores de bajo costo.

Gracias a este protocolo, los dispositivos que lo implementan pueden comunicarse entre ellos cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión lo permite. Estos dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" en referencia a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras.

| Clase | Potencia máxima permitida (mW) | Potencia máxima permitida (dBm) | Rango (Aproximado) |
|---|---|---|---|
| **Clase1** | 100 mW | 20 dBm | ~ 100 metros |
| **Clase2** | 2.5 mW | 4 dBm | ~ 10 metros |
| **Clase3** | 1 mW | 0 dBm | ~ 1 metros |

**Tabla [3], Clases de Bluetooth**
**Fuente:** http://es.wikipedia.org/wiki/Bluetooth

En la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Esto es así gracias a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte la mayor sensibilidad del dispositivo de clase 1 permite recibir la señal del otro pese a ser más débil.

### 3.1.1.2 Infrarrojo

La radiación infrarroja, radiación térmica o radiación IR es un tipo de radiación electromagnética de mayor longitud de onda que la luz visible, pero menor que la de las microondas. Consecuentemente, tiene menor frecuencia que la luz visible y mayor que las microondas. Su rango de longitudes de onda va desde unos 0,7 hasta los 100 micrómetros.1 La radiación infrarroja es emitida por cualquier cuerpo cuya temperatura sea mayor que 0 Kelvin, es decir, −273,15 grados Celsius (cero absoluto).

El nombre de infrarrojo significa por debajo del rojo pues su comienzo se encuentra adyacente al color rojo del espectro visible.

Los infrarrojos se pueden categorizar en:

- infrarrojo cercano (0,78-1,1 µm)
- infrarrojo medio (1,1-15 µm)
- infrarrojo lejano (15-100 µm)

La materia, por su caracterización energética (véase cuerpo negro) emite radiación. En general, la longitud de onda donde un cuerpo emite el máximo de radiación es inversamente proporcional a la temperatura de éste (Ley de Wien). De esta forma la mayoría de los objetos a temperaturas cotidianas tienen su máximo de emisión en el infrarrojo. Los seres vivos, en especial los mamíferos, emiten una gran proporción de radiación en la parte del espectro infrarrojo, debido a su calor corporal.

Los infrarrojos se utilizan en los equipos de visión nocturna cuando la cantidad de luz visible es insuficiente para ver los objetos. La radiación se recibe y después se refleja en una pantalla. Los objetos más calientes se convierten en los más luminosos.

Un uso muy común es el que hacen los comandos a distancia (telecomandos o mando a distancia) que generalmente utilizan los infrarrojos en vez de ondas de radio ya que no interfieren con otras señales como las señales de televisión. Los infrarrojos también se utilizan para comunicar a corta distancia los ordenadores con sus periféricos. Los aparatos que utilizan este tipo de comunicación cumplen generalmente un estándar publicado por Infrared Data Association.

La luz utilizada en las fibras ópticas es generalmente de infrarrojos.

#### 3.1.1.3 Emisores de infrarrojo industriales

Otra de las muchas aplicaciones de la radiación infrarroja es la del uso de equipos emisores de infrarrojo en el sector industrial. En este sector las aplicaciones ocupan una extensa lista pero se puede destacar su uso en aplicaciones como el secado de pinturas o barnices, secado de papel, termofijación de plásticos, precalentamiento de soldaduras, curvatura, templado y laminado del vidrio, entre otras.

La irradiación sobre el material en cuestión puede ser prolongada o momentánea teniendo en cuenta aspectos como la distancia de los emisores al material, la velocidad de paso del material (en el caso de cadenas de producción) y la temperatura que se desee conseguir.

Generalmente, cuando se habla de equipos emisores de infrarrojo, se distinguen cuatro tipos en función de la longitud de onda que utilicen:

- Emisores de infrarrojo de onda corta.
- Emisores de infrarrojo de onda media rápida
- Emisores de infrarrojo de onda media
- Emisores de infrarrojo de onda larga

#### 3.1.1.4 Zigbee

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (wireless personal área network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

En principio, el ámbito donde se prevé que esta tecnología cobre más fuerza es en domótica, como puede verse en los documentos de la ZigBee Alliance, en las referencias bibliográficas que se dan más abajo en el documento «ZigBee y Domótica». La razón de ello son diversas características que lo diferencian de otras tecnologías:

- Su bajo consumo.
- Su topología de red en malla.
- Su fácil integración (se pueden fabricar nodos con muy poca electrónica).

La siguiente tabla hace una comparación de las tecnologías inalámbricas descritas previamente para presentar sus ventajas, desventajas y así posteriormente realizar la selección del mejor requerimiento.

| Tecnología | Ventajas | Desventajas |
|---|---|---|
| **Bluetooth** | No requiere línea de vista<br>Los dispositivos pueden estar en movimiento al momento de comunicarse<br>Permite la generación de redes<br>Mayor Alcance de transmisión a diferencia de otras tecnologías<br>Realizar el registro y descubrimiento de los servicios disponibles en la red. | Mayor uso de batería en nodo visible<br>Velocidad de transmisión muy lenta para transferencia de archivos pesados.<br>Limitación entre la cantidad de periféricos que podemos usar.<br>Seguridad |
| **Infrarrojo** | Utilizada en la mayoría de los dispositivos electrónicos.<br>Alta seguridad | Requiere línea de vista<br>Las frecuencias de la banda de infrarrojo no permiten la penetración a través de paredes u obstáculos. |
| **Zigbee** | No requiere línea de vista<br>Largo alcance especialmente en sus módulos Xbee Pro 1500m<br>Bajo consumo de energía<br>Larga vida útil<br>Seguridad<br>Varias topologías<br>Un gran número de nodos dentro de sus redes.<br>Escalamiento de red<br>Agilidad de frecuencia<br>128 bits AES de cifrado<br>Económicos<br>Ideal para conexiones punto a punto y punto a multipunto<br>Opera en la banda libre de ISM 2.4Ghz para conexiones inalámbricas.<br>Reduce tiempos de espera en el envío y recepción de paquetes.<br>Baja ciclo de trabajo - Proporciona larga duración de la batería.<br>Soporte para múltiples topologías de red: Estática, dinámica, estrella y malla. | Tasa de transferencia es baja<br>Solo manipula información corta.<br>No es compatible con bluetooth debido a su baja tasa de transferencia. |

**Tabla [4], Ventajas y desventajas Tecnologías Inalámbricas**
**Elaborado:** *Autores de la Tesis*

Después de realizar el análisis de las ventajas y desventajas se procede con el estudio de Factibilidad de las mismas.

### 3.2 Estudio de Factibilidad Bluetooth – Infrarrojo – Zigbee

A continuación, se detallará los posibles factores de riesgo que puede incurrir en la elaboración del prototipo.

#### 3.2.1 Estudio de Factibilidad del uso de la tecnología Bluetooth

En la siguiente tabla [5], se muestra los resultados de las comparaciones de ventajas y desventajas, para analizar la factibilidad del uso de la tecnología Bluetooth.

| Solución | Puntaje /10 | Aspecto a considerar |
|---|---|---|
| **Bluetooth** | 8 | Costos |
| | 3 | Proveedores |
| | 8 | Características técnicas |
| | 7 | Soporte y garantía |
| | 10 | Tiempo de entrega |

**Tabla [5], Matriz de Evaluación de la Tecnología Bluetooth**
**Fuente**: *Autores de la Tesis*

##### 3.2.1.1 Riesgos de uso de la tecnología Bluetooth

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.2 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.4 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.2 | 2 | Redundancia |

**Tabla [6], Análisis de riesgo en la tecnología de Bluetooth**
**Fuente**: *Autores de la Tesis*

### 3.2.2 Estudio de Factibilidad de la tecnología Infrarrojo

| Solución | Puntaje /10 | Aspecto a considerarse |
|---|---|---|
| **Infrarrojo** | 9 | Costos |
| | 3 | Proveedores |
| | 7 | Características técnicas |
| | 7 | Soporte y garantía |
| | 10 | Tiempo de entrega |

**Tabla [7], Matriz de Evaluación de la Tecnología Infrarrojo**
**Fuente**: *Autores de la Tesis*

### 3.2.2.1 Riesgos del uso de la tecnología Infrarrojo

En el análisis de riesgo, se plantean los posibles factores de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.7 | 2 | Otros proveedores |
| **Marco regulatorio** | 0.2 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.4 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.2 | 2 | Redundancia |

**Tabla [8], Análisis de riesgo en la tecnología de Infrarrojo**
**Elaborado***: Autores de la Tesis*

### 3.2.3 Estudio de Factibilidad Zigbee

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Zigbee** | 6 | Costos |
| | 8 | Proveedores |
| | 10 | Características técnicas |
| | 9 | Soporte y garantía |
| | 10 | Tiempo de entrega |

**Tabla [9], Matriz de evaluación de la Tecnología Zigbee**
**Elaborado**: *Autores de la Tesis*

### 3.2.3.1 Riesgos del uso de la tecnología Zigbee

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.2 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.4 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.2 | 2 | Redundancia |

**Tabla [10], Probabilidad y riesgo del uso de la Tecnología Zigbee**
**Elaborado:** *Autores de la Tesis*

### 3.2.4 Jerarquización de Factibilidad

Los criterios usados para jerarquizar cada una de las factibilidades de cada solución alternativa. También describe el mecanismo de puntuación usado para añadir peso a los totales individuales y asignar un puntaje total para cada solución. Los criterios tomados en cuenta para jerarquizar la factibilidad de cada solución son los siguientes: Costos, proveedores, características técnicas, soporte de garantía y tiempo de entrega.

| | Bluetooth | | | Infrarrojo | | | Zigbee | | |
|---|---|---|---|---|---|---|---|---|---|
| **Criterio** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** |
| **Costos** | 8 | 0.3 | 2.4 | 9 | 0.3 | 2.7 | 6 | 0.3 | 1.8 |
| **Proveedores** | 3 | 0.1 | 0.3 | 3 | 0.1 | 0.3 | 8 | 0.1 | 0.8 |
| **Características técnicas** | 8 | 0.4 | 3.2 | 7 | 0.4 | 2.8 | 10 | 0.4 | 4 |
| **Soporte y garantía** | 7 | 0.1 | 0.7 | 7 | 0.1 | 0.7 | 9 | 0.1 | 0.9 |
| **Tiempo de entrega** | 10 | 0.1 | 1 | 10 | 0.1 | 1 | 10 | 0.1 | 1 |
| **Puntaje Total** | | | **7.6** | | | **7.5** | | | **8.5** |

**Tabla [11], Cuadro comparativo de Puntajes**
**Elaborado:** *Autores de la Tesis*

### 3.2.5 Análisis del cuadro comparativo de Puntajes

Basados en la tabla anterior, se identifico como solución ganadora, en base al análisis de las calificaciones, a la solución Zigbee con la cual usará en el proyecto de tesis por sus múltiples ventajas detalladas.

## 3.3 Análisis de Software

Se analizará los diferentes programas con el cual será implementado el sistema.

### 3.3.1 Lenguajes de programación para FPGA

#### 3.3.1.1 Concepto de Lenguaje VHDL

VHDL es el acrónimo que representa la combinación de VHSIC y HDL, donde VHSIC es el acrónimo de Very High Speed Integrated Circuit y HDL es a su vez el acrónimo de Hardware Description Language.

Es un lenguaje definido por el IEEE (Institute of Electrical and Electronics Engineers) (ANSI/IEEE 1076-1993) usado por ingenieros para describir circuitos digitales. Otros métodos para diseñar circuitos son la captura de esquemas (con herramientas CAD) y los diagramas de bloques, pero éstos no son prácticos en diseños complejos. Otros lenguajes para el mismo propósito son Verilog y ABEL.

Aunque puede ser usado de forma general para describir cualquier circuito se usa principalmente para programar PLD (Programable Logic Device - Dispositivo Lógico Programable), FPGA (Field Programmable Gate Array), ASIC y similares.

#### 3.3.1.1.1 Formas de describir un circuito

Dentro del VHDL hay varias formas con las que podemos diseñar el mismo circuito y es tarea del diseñador elegir la más apropiada.

Funcional: Describimos la forma en que se comporta el circuito. Esta es la forma que más se parece a los lenguajes de software ya que la descripción es secuencial. Estas sentencias secuenciales se encuentran dentro de los llamados procesos en VHDL. Los procesos son ejecutados en paralelo entre sí, y en paralelo con asignaciones concurrentes de señales y con las instancias a otros componentes.

- Flujo de datos: describe asignaciones concurrentes (en paralelo) de señales.
- Estructural: se describe el circuito con instancias de componentes. Estas instancias forman un diseño de jerarquía superior, al conectar los puertos de estas instancias con las señales internas del circuito, o con puertos del circuito de jerarquía superior.
- Mixta: combinación de todas o algunas de las anteriores.

En VHDL también existen formas metódicas para el diseño de máquinas de estados, filtros digitales, bancos de pruebas etc.

#### 3.3.1.1.2 Secuencia de diseño

El flujo de diseño de un sistema puede ser:

- División del diseño principal en módulos separados. La modularidad es uno de los conceptos principales de todo diseño. Normalmente se diferencia entre dos metodologías de diseño: top-down y botton-up. La metodología top-down consiste en que un diseño complejo se divide en diseños más sencillos que se puedan diseñar (o describir) más fácilmente. La metodología botton-up consiste en construir un diseño complejo a partir de módulos, ya diseñados, más simples. En la práctica, un diseño usa generalmente ambas metodologías.
- Entrada de diseños, pueden usarse diversos métodos tal como VHDL como se ve anteriormente.
- Simulación funcional, es decir, comprobaremos que lo escrito en el punto anterior realmente funciona como queremos, si no lo hace tendremos que modificarlo. En este tipo de simulación se comprueba que el código VHDL o Verilog (u otro tipo de lenguaje HDL) ejecuta correctamente lo que se pretende.
- Síntesis. En este paso se adapta el diseño anterior (que sabemos que funciona) a un hardware en concreto, ya sea una FPGA o un ASIC. Hay sentencias del lenguaje que no son sintetizables, como por ejemplo divisiones o exponenciaciones con números no constantes. El hecho de que no todas las expresiones en VHDL sean sintetizables es que el VHDL es un lenguaje genérico para modelado de sistemas (no sólo para diseño de circuitos digitales), por lo que hay expresiones que no pueden ser transformadas a circuitos digitales. Durante la síntesis se tiene en cuenta la estructura interna del dispositivo, y se definen restricciones, como la asignación de pines. El sintetizador optimiza las expresiones lógicas con objeto de que ocupen menor área, o bien son eliminadas las expresiones lógicas que no son usadas por el circuito.
- Simulación post-síntesis. En este tipo de simulación se comprueba que el sintetizador ha realizado correctamente la síntesis del circuito, al transformar el código HDL en bloques lógicos conectados entre sí. Este paso es necesario ya

que, a veces, los sintetizadores producen resultados de síntesis incorrectos, o bien realiza simplificaciones del circuito al optimizarlo.

- Ubicación y enrutamiento. El proceso de ubicación consiste en situar los bloques digitales obtenidos en la síntesis de forma óptima, de forma que aquellos bloques que se encuentran muy interconectados entre sí se sitúen próximamente. El proceso de enrutamiento consiste en interconectar adecuadamente los bloques entre sí, intentando minimizar retardos de propagación para maximizar la frecuencia máxima de funcionamiento del dispositivo.
- Anotación final. Una vez ha sido completado el proceso de ubicación y enrutamiento, se extraen los retardos de los bloques y sus interconexiones, con objeto de poder realizar una simulación temporal (también llamada simulación post-layout). Estos retardos son anotados en un fichero SDF (Standard Delay Format) que asocia a cada bloque o interconexión un retardo mínimo/típico/máximo.
- Simulación temporal. A pesar de la simulación anterior puede que el diseño no funcione cuando se programa, una de las causas puede ser por los retardos internos del chip. Con esta simulación se puede comprobar, y si hay errores se tiene que volver a uno de los anteriores pasos.
- Programación en el dispositivo. Se implementa el diseño en el dispositivo final y se comprueba el resultado.

#### 3.3.1.2 Concepto de Lenguaje Verilog

Verilog es un lenguaje de descripción de hardware (HDL, del Inglés Hardware Description Language) usado para modelar sistemas electrónicos. El lenguaje, algunas veces llamado Verilog HDL, soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes niveles de abstracción.

Los diseñadores de Verilog querían un lenguaje con una sintaxis similar a la del lenguaje de programación C, de tal manera que le resultara familiar a los ingenieros y así fuera rápidamente aceptada. El lenguaje tiene un preprocesador como C, y la mayoría de

palabras reservadas de control como "if", "while", etc, son similares. El mecanismo de formateo en las rutinas de impresión y en los operadores del lenguaje (y su precedencia) son también similares.

A diferencia del lenguaje C, Verilog usa Begin/End en lugar de llaves para definir un bloque de código. Por otro lado la definición de constantes en Verilog requiere la longitud de bits con su base. Verilog no tiene estructuras, apuntadores o funciones recursivas. Finalmente el concepto de tiempo, muy importante en un HDL, no se encuentra en C.

El lenguaje difiere de los lenguajes de programación convencionales, en que la ejecución de las sentencias no es estrictamente lineal. Un diseño en Verilog consiste de una jerarquía de módulos. Los módulos son definidos con conjuntos de puertos de entrada, salida y bidireccionales. Internamente un módulo contiene una lista de cables y registros. Las sentencias concurrentes y secuenciales definen el comportamiento del módulo, describiendo las relaciones entre los puertos, cables y registros. Las sentencias secuenciales son colocadas dentro de un bloque begin/end y ejecutado en orden secuencial, pero todas las sentencias concurrentes y todos los bloques begin/end son ejecutados en paralelo en el diseño. Un módulo puede contener una o más instancias de otro módulo para definir un sub-comportamiento.

Un subconjunto de sentencias en el lenguaje es sintetizable. Si los módulos en un diseño contienen sólo sentencias sintetizables, se puede usar software para convertir o sintetizar el diseño en una lista de nodos que describe los componentes básicos y los conectores que deben implementarse en hardware. La lista de nodos puede entonces ser transformada en una forma describiendo las celdas estándar de un circuito integrado, por ejemplo ASIC, o una cadena de bits para un dispositivo de lógica programable (PLD) como puede ser una FPGA o un CPLD.

En la siguiente tabla, se analiza las ventajas y desventajas de los lenguajes de programación para su posterior comparación y selección.

| Lenguaje | Ventajas | Desventajas |
|---|---|---|
| **Verilog** | Los tipos de datos son más simples y están orientados al modelamiento en hardware.<br>Permite describir módulos en términos de compuertas Flip-Flops lo cual permite llegar a niveles muy bajos de abstracción | La compilación es un medio de acelerar la simulación, pero no ha cambiado la naturaleza original de la lengua. |
| **VHDL** | Se permite el uso de tipos de datos definidos por el lenguaje y por el usuario.<br>Diseño de múltiples unidades (entidad pares de arquitectura), que residen en el mismo sistema de archivos, puede ser compilado por separado si así lo desea.<br>VHDL permite diseñar, modelar, y comprobar un sistema desde un alto nivel de abstracción bajando hasta el nivel de definición estructural de puertas.<br>Al estar basado en un estándar (IEEE Std 1076-1987) los ingenieros de toda la industria de diseño pueden usar este lenguaje para minimizar errores de comunicación y problemas de compatibilidad. | Pueden generarse latches[27] no deseados, p.ej. cuando todas las opciones de una sentencia condicional no están especificadas. |

**Tabla [12], Ventajas y desventajas de los Lenguajes de Programación**
**Elaborado:** *Autores de la Tesis*

[27] Un latch (lat memori inglet) es un circuito electrónico usado para almacenar información en sistemas lógicos asíncronos.

### 3.3.2 Factibilidad de uso de los Lenguajes de Programación Verilog & VHDL

A continuación, se analizarán los lenguajes de programación Verilog y VHDL, mediante matrices comparativas para determinar cuál es el lenguaje más apropiado para utilizar en el prototipo.

#### 3.3.2.2 Verilog

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Verilog** | 8 | Costos |
| | 2 | Proveedores |
| | 8 | Características técnicas |
| | 5 | Soporte y garantía |
| | 5 | Tiempo de entrega |

**Tabla [13], Matriz de evaluación del lenguaje Verilog**
**Elaborado:** *Autores de la Tesis*

#### 3.3.2.2.1 Riesgos de uso del lenguaje Verilog

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.1 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.1 | 3 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 1 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [14], Probabilidad e impacto de riesgo del uso del lenguaje Verilog**
**Elaborado:** *Autores de la Tesis*

#### 3.3.2.3 VHDL

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **VHDL** | 8 | Costos |
| | 3 | Proveedores |
| | 9 | Características técnicas |
| | 6 | Soporte y garantía |
| | 5 | Tiempo de entrega |

**Tabla [15], Matriz de evaluación del lenguaje VHDL**
**Elaborado:** *Autores de la Tesis*

##### 3.3.2.3.1 Riesgos del uso del lenguaje VHDL

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.2 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.4 | 4 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 1 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.2 | 1 | Redundancia |

**Tabla [16], Probabilidad y riesgo del uso del lenguaje VHDL**
**Elaborado:** *Autores de la Tesis*

### 3.3.3 Jerarquización de Factibilidad

Los criterios usados para jerarquizar cada una de las factibilidades de cada solución alternativa. También describe el mecanismo de puntuación usado para añadir peso a los totales individuales y asignar un puntaje total para cada solución.

Los criterios tomados en cuenta para jerarquizar la factibilidad de cada solución son los siguientes: Costos, proveedores, características técnicas, soporte de garantía y tiempo de entrega

| | Verilog | | | VHDL | | |
|---|---|---|---|---|---|---|
| **Criterio** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** |
| **Costos** | 8 | 0.1 | 0.8 | 8 | 0.1 | 0.8 |
| **Proveedores** | 2 | 0.4 | 0.8 | 3 | 0.7 | 2.1 |
| **Características técnicas** | 8 | 0.5 | 4 | 9 | 0.8 | 7.2 |
| **Soporte y garantía** | 5 | 0.7 | 3.5 | 6 | 0.8 | 4.8 |
| **Tiempo de entrega** | 5 | 0.8 | 4 | 5 | 0.9 | 4.5 |
| **Puntaje Total** | | | **13.1** | | | **19.4** |

**Tabla [17], Cuadro comparativo de puntajes**
**Elaborado:** *Autores de la Tesis*

### 3.3.4 Análisis del cuadro comparativo de puntajes

Basados en la tabla anterior, se identifico como solución ganadora, y tomando en cuenta los objetivos que se persigue, en base al análisis de las calificaciones, al lenguaje de

programación VHDL el cual usará en el proyecto de tesis por sus múltiples ventajas detalladas.

## 3.4 LENGUAJE DE PROGRAMACIÓN DE MICROCONTROLADORES

Existen varios lenguajes de programación así como también compiladores de microcontroladores unos más avanzados y con más funciones pero todos tienen la misma finalidad, que es procesar la información.

Se describe a continuación los siguientes lenguajes.

### 3.4.1 Ventajas y Desventajas del uso del Lenguaje Basic

**Ventajas**

- Lenguaje de fácil manejo muy simple y con instrucciones fácilmente legibles.

**Desventajas**

- Complicado manejo de interrupciones simultaneas.
- Tiene limitaciones al generar el archivo .hex por este motivo no optimiza el tamaño del programa con relación a la memoria del PIC.
- Únicamente trabaja bajo ambiente Windows.

### 3.4.2 Ventajas y Desventajas del uso del Lenguaje C

**Ventajas**

- Lenguaje de alto nivel más cercano a la maquina
- Construcción de rutinas matemáticas fácilmente.
- Compatible con ensamblador sobre todo en PICs de gama alta.

- Creación de macros con este lenguaje, para después simplificar el código.

**Desventajas**

- La compilación de los programas resulta en muchos casos muy extenso y pesado, es por este motivo que se debe tener en cuenta la capacidad de la memoria del PIC que se está programando.

### 3.4.3 Ventajas y Desventajas del uso del Lenguaje Ensamblador

**Ventajas**

- Es el lenguaje de bajo nivel de los PICs tanto para gama baja, media y alta.
- Eficiente aprovechamiento de los recursos del PIC.
- También permite la creación de macros.
- Controla los tiempos y los registros bit a bit.
- Controla muy bien las interrupciones simultáneas.
- Al generar el archivo .hex este es completamente optimizado.

**Desventajas**

- Requiere conocimientos avanzados en programación.
- Código extenso.

## 3.5 Estudio de Factibilidad de uso de los Lenguaje de programación para microcontroladores

A continuación se analizarán los lenguajes de programación para microcontroladores, mediante matrices comparativas para determinar cuál es el lenguaje más apropiado para utilizar en el prototipo.

### 3.5.1 Lenguaje Basic

| Lenguaje | Puntaje /10 | Aspecto a considerarse |
|---|---|---|
| **Basic** | 7 | Costos |
| | 3 | Proveedores |
| | 8 | Características técnicas |
| | 7 | Soporte y garantía |
| | 3 | Tiempo de entrega |

**Tabla [18], Matriz de evaluación del lenguaje Basic**
**Elaborado:** *Autores de la Tesis*

### 3.5.1 Riesgos del uso del lenguaje Basic

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.7 | 4 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [19], Probabilidad e Impacto de riesgo del uso del lenguaje Basic**
**Elaborado:** *Autores de la Tesis*

### 3.6 Lenguaje C

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **C** | 8 | Costos |
| | 10 | Proveedores |
| | 9 | Características técnicas |
| | 9 | Soporte y garantía |
| | 10 | Tiempo de entrega |

**Tabla [20], Matriz de evaluación del lenguaje C**
**Elaborado:** *Autores de la Tesis*

#### 3.6.1 Riesgos del uso del lenguaje C

En el análisis de riesgo, se plantean los posibles factores de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.7 | 4 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [21], Probabilidad e impacto de riesgo del uso del lenguaje C**
**Elaborado:** *Autores de la Tesis*

### 3.7 Lenguaje Ensamblador

| Solución | Puntaje /10 | Aspectos a considerarse |
|---|---|---|
| **Ensamblador** | 3 | Costos |
| | 3 | Proveedores |
| | 6 | Características técnicas |
| | 5 | Soporte y garantía |
| | 4 | Tiempo de entrega |

**Tabla [22], Estudio de Factibilidad de Lenguaje Ensamblador**
**Elaborado:** *Autores de la Tesis*

#### 3.7.1 Riesgos del uso del lenguaje Ensamblador

En el análisis de riesgo, se plantean los posibles factores de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.7 | 4 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [23], Probabilidad e Impacto de riesgo en el uso del lenguaje Ensamblador**
**Elaborado:** *Autores de la Tesis*

### 3.7.2 Jerarquización de Factibilidad

Los criterios usados para jerarquizar cada una de las factibilidades de cada solución alternativa. También describe el mecanismo de puntuación usado para añadir peso a los totales individuales y asignar un puntaje total para cada solución.

Los criterios tomados en cuenta para jerarquizar la factibilidad de cada solución son los siguientes: Costos, proveedores, características técnicas, soporte de garantía y tiempo de entrega

| | **Basic** | | | **C** | | | **Ensamblador** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Criterio** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** |
| **Costos** | 7 | 0.3 | 2.1 | 8 | 0.3 | 2.7 | 3 | 0.3 | 0.9 |
| **Proveedores** | 3 | 0.1 | 0.3 | 10 | 0.1 | 0.3 | 3 | 0.1 | 0.3 |
| **Características técnicas** | 8 | 0.4 | 3.2 | 9 | 0.4 | 2.8 | 6 | 0.4 | 2.4 |
| **Soporte y garantía** | 7 | 0.1 | 0.7 | 9 | 0.1 | 0.7 | 5 | 0.1 | 0.5 |
| **Tiempo de entrega** | 3 | 0.1 | 0.3 | 10 | 0.1 | 1 | 4 | 0.1 | 0.4 |
| **Puntaje Total** | | | **6.6** | | | **8.9** | | | **4.5** |

**Tabla [24], Cuadro comparativo de puntajes**
**Elaborado:** *Autores de la Tesis*

### 3.7.3 Análisis del cuadro comparativo de puntajes

Basados en la tabla anterior, se identifico como solución ganadora, y tomando en cuenta los objetivos que se persigue, en base al análisis de las calificaciones, al lenguaje de programación para microcontroladores C el cual usará en el proyecto de tesis por sus múltiples ventajas detalladas.

### 3.8 Herramientas de desarrollo Gráfico

Existen varias herramientas para la obtención de datos dentro de nuestro proyecto de tesis, a continuación se analizará las más populares.

#### 3.8.1 Eclipse

Plataforma Open Source[28] de desarrollo para aplicaciones basadas en Java y otros lenguajes.

#### 3.8.2 Netbeans

Herramienta para programadores pensada para escribir, compilar, depurar, ejecutar programas Java agradable y fácil manipulación de su entorno gráfico.

#### 3.8.3 Sharp Developer

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET, similar al de Java aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi).

Para la obtención de datos y muestra de los mismos se usará el lenguaje de programación C#, ya que el mismo es de dominio para los desarrolladores de la tesis.

[28] Es el software que está licenciado de tal manera que los usuarios pueden estudiar, modificar y mejorar su diseño mediante la disponibilidad de su código fuente.

### 3.9 Estudio de Factibilidad de las Herramientas de Desarrollo Grafico

#### 3.9.1 Herramienta Eclipse

En la siguiente tabla se muestra los resultados de las evaluaciones, se describe los métodos utilizados por la Herramienta Eclipse.

| Solución | Puntaje /10 | Aspectos a considerarse |
|---|---|---|
| **Eclipse** | 9 | Costos |
| | 9 | Proveedores |
| | 8 | Características técnicas |
| | 8 | Soporte y garantía |
| | 9 | Tiempo de entrega |

**Tabla [25], Matriz de evaluación de uso de la herramienta Eclipse**
**Elaborado:** *Autores de la Tesis*

#### 3.9.2.1 Riesgos del uso de la herramienta Eclipse

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.8 | 4 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [26], Probabilidades e Impacto de riesgo del uso de la herramienta Eclipse**
**Elaborado:** *Autores de la Tesis*

### 3.9.2 Herramienta Netbeans

En la siguiente tabla se muestra los resultados de las evaluaciones, se describe los métodos utilizados por Herramienta Netbeans.

| Solución | Puntaje /10 | Aspectos a considerar |
|---|---|---|
| **Netbeans** | 9 | Costos |
| | 9 | Proveedores |
| | 7 | Características técnicas |
| | 8 | Soporte y garantía |
| | 9 | Tiempo de entrega |

**Tabla [27], Matriz de evaluación de uso de la herramienta Netbeans**
**Elaborado:** *Autores de la Tesis*

#### 3.9.2.1 Riesgos del uso de la herramienta Netbeans

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.8 | 4 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [28], Probabilidad e impacto de riesgo con el uso de la herramienta Netbeans**
**Elaborado:** *Autores de la Tesis*

### 3.9.3 Herramienta Sharp Developer

En la siguiente tabla se muestra los resultados de las evaluaciones, se describe los métodos utilizados por la Herramienta Sharp Developer.

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Sharp Developer** | 9 | Costos |
| | 9 | Proveedores |
| | 9 | Características técnicas |
| | 8 | Soporte y garantía |
| | 9 | Tiempo de entrega |

**Tabla [29], Matriz de evaluación de uso de la herramienta Sharp Developer**
**Elaborado:** *Autores de la Tesis*

#### 3.9.3.1 Riesgos de uso de la herramienta Sharp Developer

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.8 | 4 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [30], Probabilidad e impacto de riesgo con el uso de la herramienta Sharp**
**Elaborado:** *Autores de la Tesis*

### 3.9.4 Jerarquización de Factibilidad

Describe los criterios usados para jerarquizar cada una de las factibilidades de cada solución alternativa. También describe el mecanismo de puntuación usado para añadir peso a los totales individuales y asignar un puntaje total para cada solución.

Los criterios tomados en cuenta para jerarquizar la factibilidad de cada solución son los siguientes: Costos, proveedores, características técnicas, soporte de garantía y tiempo de entrega

| | **Eclipse** | | | **Netbeans** | | | **Sharp Developer** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Criterio** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** |
| **Costos** | 9 | 0.3 | 2.1 | 9 | 0.3 | 2.7 | 9 | 0.3 | 0.9 |
| **Proveedores** | 9 | 0.1 | 0.3 | 9 | 0.1 | 0.3 | 9 | 0.1 | 0.3 |
| **Características técnicas** | 8 | 0.4 | 3.2 | 7 | 0.4 | 2.8 | 9 | 0.4 | 2.4 |
| **Soporte y garantía** | 8 | 0.1 | 0.7 | 8 | 0.1 | 0.7 | 8 | 0.1 | 0.5 |
| **Tiempo de entrega** | 9 | 0.1 | 0.3 | 9 | 0.1 | 1 | 9 | 0.1 | 0.4 |
| **Puntaje Total** | | | **8.5** | | | **8.1** | | | **8.9** |

**Tabla [31], Cuadro Comparativo de puntajes**
**Elaborado:** *Autores de la Tesis*

### 3.9.5 Análisis del cuadro comparativo de puntajes

Basados en los datos anteriores, se puede observar que las tres herramientas de desarrollo son Open Source, y tomando en cuenta los objetivos que se persigue en este proyecto de tesis se ha optado por escoger IDE Sharp Developer ya que es un lenguaje que posee características que se acoplan a este proyecto.

### 3.10 Herramientas de desarrollo Basic para Microcontroladores

#### 3.10.1 Características MikroC Pro

Es una poderosa herramienta para programar Microcontroladores mediante lenguaje Basic.

**Características**

- MicroICD – Depurador en circuito
- Gestor de proyectos: Permite a los usuarios gestionar multiples proyectos.
- Explorador de código: permite supervisar las variables, funciones, procedimientos que se están usando.
- Administrador de librerías: Permite ver las variables que se están utilizando y las librerías que son almacenadas al momento de compilar el programa.
- Asistente de código: Ahorra tiempo al momento de escribir.

#### 3.10.2 Características Microcode Studio

Es una herramienta muy potente para la programación de Microcontroladores en lenguaje Basic diseñada específicamente para MicroEngineering Labs PBP y PBP PRO.

El editor principal de esta herramienta proporciona la síntesis completa y resalta el código con ayuda conceptual.

Tiene un explorador de código que permite incluir archivos, constantes, variables, modificadores y etiquetas que están dentro del código fuente que se genera, de igual forma posee características de búsqueda y reemplazo. Contiene una ventana de mensajes de error basta con hacer clic sobre el error y automáticamente el Microcode Studio te llevara a la línea de código donde se encuentra el error, también tiene una ventana de comunicaciones que permite depurar y ver la ventana del microcontrolador.

### 3.11 Estudio de Factibilidad del uso de herramientas para Microcontroladores

#### 3.11.1 Herramienta MikroC Pro

En la siguiente tabla, se muestra los resultados de las evaluaciones, se describe los métodos utilizados por la Herramienta MicroC Pro.

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **MikroC Pro** | 9 | Costos |
| | 7 | Proveedores |
| | 8 | Características técnicas |
| | 8 | Soporte y garantía |
| | 8 | Tiempo de entrega |

**Tabla [32], Matriz de evaluación de uso de la herramienta MikroC Pro**
**Elaborado:** *Autores de la Tesis*

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

##### 3.11.1.1 Riesgos del uso de la herramienta MicroC Pro

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.6 | 4 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.8 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [33], Probabilidad e impacto de riesgo con el uso de la herramienta MicroC**
**Elaborado:** *Autores de la Tesis*

### 3.11.2 Herramienta Microcode Studio

En la siguiente tabla, se muestra los resultados de las evaluaciones, se describe los métodos utilizados por Herramienta la Herramienta Microcode Studio.

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Microcode Studio** | 8 | Costos |
| | 4 | Proveedores |
| | 6 | Características técnicas |
| | 7 | Soporte y garantía |
| | 7 | Tiempo de entrega |

**Tabla [34],**
**Matriz de evaluación de uso de la herramienta Microcode Studio**
**Elaborado:** *Autores de la Tesis*

#### 3.11.2.1 Riesgos del uso de la herramienta Microcode Studio

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo.

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.6 | 4 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.8 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.2 | 2 | Replanteo de costos |
| **Factores ambientales** | 0.1 | 1 | Plan de contingencia |
| **Desastres** | 0.1 | 1 | Redundancia |

**Tabla [35]**
**Probabilidad e impacto de riesgo con el uso de la herramienta Microcode Studio**
**Elaborado:** *Autores de la Tesis*

### 3.11.3 Jerarquización de Factibilidad

Los criterios tomados en cuenta para jerarquizar la factibilidad de cada tecnología son los siguientes: Costos, proveedores, características técnicas, soporte de garantía y tiempo de entrega

| | Mikro C Pro | | | Microcode Studio | | |
|---|---|---|---|---|---|---|
| **Criterio** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** |
| **Costos** | 9 | 0.3 | 2.1 | 8 | 0.3 | 2.7 |
| **Proveedores** | 7 | 0.1 | 0.3 | 4 | 0.1 | 0.3 |
| **Características técnicas** | 8 | 0.4 | 3.2 | 6 | 0.4 | 2.8 |
| **Soporte y garantía** | 8 | 0.1 | 0.7 | 7 | 0.1 | 0.7 |
| **Tiempo de entrega** | 8 | 0.1 | 0.3 | 7 | 0.1 | 1 |
| **Puntaje Total** | | | **8.2** | | | **6.6** |

**Tabla [36], Cuadro comparativo de puntajes**
**Elaborado:** *Autores de la Tesis*

### 3.11.4 Resultados de Factibilidad

Del estudio de las herramientas para la programación de los microcontroladores se ha optado por la herramienta de desarrollo MikroC Pro con lo cual será programado el microcontrolador.

## 3.12 Análisis de Hardware

Se realizara el análisis de los dispositivos que serán implementados dentro del proyecto de tesis.

### 3.12.1 FPGA

#### 3.12.1.1 Proveedores

Existen dos productores de FPGA de procedencia internacional, y en el Ecuador no existe ninguno.

- Xilinx y Altera son los productores de FPGA.
- Altera es el otro gran líder.
- Lattice Semiconductor lanzó al mercado dispositivos FPGA con tecnología de 90nm. En adición, Lattice es un proveedor líder en tecnología no volátil, FPGA basadas en tecnología Flash, con productos de 90nm y 130nm.
- Actel tiene FPGAs basados en tecnología Flash reprogrammable. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.
- QuickLogic tiene productos basados en antifusibles (programables una sola vez).
- Atmel es uno de los fabricantes cuyos productos son reconfigurables (el Xilinx XC62xx fue uno de estos, pero no están siendo fabricados actualmente). Ellos se enfocaron en proveer microcontroladores AVR con FPGAs, todo en el mismo encapsulado.
- Achronix Semiconductor tienen en desarrollo FPGAs muy veloces. Planean sacar al mercado a comienzos de 2007 FPGAs con velocidades cercanas a los 2GHz.
- MathStar, Inc. ofrecen FPGA que ellos llaman FPOA (Arreglo de objetos de matriz programable).

Se detallará a las tres mejores empresas distribuidores de FPGA.

**Xilinx**

Xilinx es la mayor empresa en investigación y desarrollo de chips conocidos como field-programmable gate arrays (FPGAs).

**Altera**

Altera Corporation (NASDAQ: ALTR) es un fabricante líder de dispositivos lógicos programables. Es un miembro del grupo NASDAQ-100y del S&P 500.

Altera es uno de los pioneros de la lógica programable, siguiendo líderes notables anteriores como Signetics y MMI en la introducción de PLDs. Altera desarrolla algunas características que están orientadas hacia capacidad de sistemas en chips programables (SOPC). Algunos de los ejemplos más recientes incluyen memoria embebida, procesadores embebidos, y transceptores de alta velocidad. El éxito en lanzamientos de productos de 130nm y 90nm son buenos casos de estudio. Los procesadores soft-core Nios II y Nios de Altera y los dispositivos HardCopy II y HardCopy están extendiendo el alcance de Altera en el mercado, y coloca a esta empresa en el mundo de los procesadores embebidos y ASICs estructuradas respectivamente. Entre sus principales competidores están: Xilinx, Lattice Semiconductor, Actel, Quicklogic y Atmel.

**Atmel**

Atmel sirve a los mercados de la electrónica de consumo, comunicaciones, computadores, redes, electrónica industrial, equipos médicos, automotriz, aeroespacial y militar. Es una industria líder en sistemas seguros, especialmente en el mercado de las tarjetas seguras.

A continuación se realiza el estudio de factibilidad de las tres empresas distribuidoras del entrenador Spartan 3E.

**3.12.1.1.1 Análisis de proveedores del Entrenador FPGA 3E**

**3.12.1.1.1.2 Empresa Xilinx**

En la siguiente tabla se muestra los resultados de las evaluaciones, se describe los métodos utilizados por la empresa Xilinx.

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Xilinx** | 9 | Costos |
| | 7 | Proveedores |
| | 9 | Características técnicas |
| | 5 | Soporte y garantía |
| | 5 | Tiempo de entrega |

**Tabla [37], Matriz de evaluación de la empresa Xilinx**
**Elaborado:** *Autores de la Tesis*

#### 3.12.1.1.1.2.1 Riesgos de adquirir a la empresa Xilinx.

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.9 | 5 | Replanteo de costos |
| **Factores ambientales** | 0.5 | 2 | Plan de contingencia |
| **Desastres** | 0.5 | 2 | Redundancia |

**Tabla [38], Probabilidad e impacto de riesgo con la empresa Xilinx**
**Elaborado:** *Autores de la Tesis*

#### 3.12.1.1.3 Empresa Altera

En la siguiente tabla se muestra los resultados de las evaluaciones, se describe los métodos utilizados por la Empresa Altera.

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Altera** | 7 | Costos |
| | 5 | Proveedores |
| | 9 | Características técnicas |
| | 7 | Soporte y garantía |
| | 6 | Tiempo de entrega |

**Tabla [39], Matriz de evaluación de la empresa Altera**
**Elaborado:** *Autores de la Tesis*

#### 3.12.1.1.3.1 Riesgos de adquirir a la empresa Altera

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.9 | 5 | Replanteo de costos |
| **Factores ambientales** | 0.5 | 2 | Plan de contingencia |
| **Desastres** | 0.5 | 2 | Redundancia |

**Tabla [40], Probabilidad de impacto y riesgo con la empresa Altera**
**Elaborado:** *Autores de la Tesis*

#### 3.12.1.1.4 Empresa Atmel

En la siguiente tabla se muestra los resultados de las evaluaciones, se describe los métodos utilizados por la Empresa Atmel.

| Solución | Puntaje /10 | Método de Evaluación |
|---|---|---|
| **Atmel** | 6 | Costos |
| | 4 | Proveedores |
| | 9 | Características técnicas |
| | 6 | Soporte y garantía |
| | 5 | Tiempo de entrega |

**Tabla [41], Matriz de evaluación Empresa Atmel**
**Elaborado:** *Autores de la Tesis*

#### 3.12.1.1.4.1 Riesgos de adquirir a la empresa Atmel

En el análisis de riesgo, se plantean los posibles factores evaluados para la probabilidad de riesgo en factores de 1 y para el impacto en factores de 1 a 5, siendo 5 el valor más significativo

| Descripción del Riesgo | Probabilidad del Riesgo | Impacto del Riesgo /5 | Acciones Requeridas para Mitigar el Riesgo |
|---|---|---|---|
| **Incumplimientos** | 0.9 | 5 | Otros proveedores |
| **Marco regulatorio** | 0.7 | 3 | Modificaciones técnicas |
| **Aceptación de la comunidad** | 0.9 | 5 | Negociación, consensuar |
| **Políticas económicas** | 0.9 | 5 | Replanteo de costos |
| **Factores ambientales** | 0.5 | 2 | Plan de contingencia |
| **Desastres** | 0.5 | 2 | Redundancia |

**Tabla [42], Probabilidad de impacto y riesgo con la empresa Atmel**
**Elaborado:** *Autores de la Tesis*

### 3.12.2 Jerarquización de Factibilidad

Describe los criterios usados para jerarquizar cada una de las factibilidades de cada solución alternativa. También describe el mecanismo de puntuación usado para añadir peso a los totales individuales y asignar un puntaje total para cada solución.

Los criterios tomados en cuenta para jerarquizar la factibilidad de cada solución son los siguientes: Costos, proveedores, características técnicas, soporte de garantía y tiempo de entrega.

| | Eclipse | | | Netbeans | | | Sharp Developer | | |
|---|---|---|---|---|---|---|---|---|---|
| **Criterio** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** | **Puntaje** | **Peso** | **Total** |
| **Costos** | 9 | 0.3 | 2.1 | 7 | 0.3 | 2.7 | 6 | 0.3 | 0.9 |
| **Proveedores** | 7 | 0.1 | 0.3 | 5 | 0.1 | 0.3 | 4 | 0.1 | 0.3 |
| **Características técnicas** | 9 | 0.4 | 3.2 | 9 | 0.4 | 2.8 | 9 | 0.4 | 2.4 |
| **Soporte y garantía** | 5 | 0.1 | 0.7 | 7 | 0.1 | 0.7 | 6 | 0.1 | 0.5 |
| **Tiempo de entrega** | 5 | 0.1 | 0.3 | 6 | 0.1 | 1 | 5 | 0.1 | 0.4 |
| **Puntaje Total** | | | **8.0** | | | **7.5** | | | **6.9** |

**Tabla [43], Cuadro comparativo de puntajes**
**Elaborado:** *Autores de la Tesis*

### 3.12.3 Análisis del cuadro comparativo

Basados en los datos anteriores, se puede observar que las tres empresas distribuyen el entrenador de FPGA, tomando en cuenta los objetivos que se persigue para este proyecto de tesis, se ha optado por adquirir a la empresa Xilinx.

## 3.13 Módulos Xbee

Una de las empresas líderes en la fabricación de los módulos Xbee es Digi que se distribuye en el mundo con oficinas en América del Norte, Europa y Asia con aproximadamente 262 distribuidores en 70 países convirtiéndola en la empresa más grande en la fabricación de módulos Zigbee se refiere, y con un sin número de dispositivos electrónicos que adoptan también esta tecnología.

Otra empresa que a lo largo de estos últimos años se ha visto comprometida con las necesidades que exige el mercado en relación a módulos Zigbee y redes inalámbricas se

refiere es Microchip que ha lanzado su gama de nuevos productos basados en Zigbee y se espera que sea una de las competidoras de Digi a pesar que esta última ya lleva una gran ventaja.

Otro factor a tomar en cuenta son las dos series que ofrecen los módulos Zigbee serie 1 y serie 2 cada unos con sus propias características cabe señalar que estas dos series no son compatibles entre ellas así que se recomienda tener cuidado al momento de utilizar.

En la siguiente tabla, se definen las características de los módulos serie 1 y serie 2

| | Xbee serie 1 | Xbee serie 2 |
|---|---|---|
| Alcance | 100 ft (30m) | 133 ft (40m) |
| Potencia de Salida | 1 mW (odbm) | 2 mW (odbm) |
| Transmisión | 250 kbps | 250 kbps |
| Voltaje de Alimentación | 2.8 a 3.4 v | 2.8 a 3.6 v |
| Consumo RX | 45 mA | 40 mA |
| Consumo TX | 50 mA | 40 mA |
| Frecuencia | ISM 2.4 Ghz | ISM 2.4 Ghz |
| Dimensiones | 0.0960” x 1.087” | 0.0960” x 1.087” |
| Temperatura de Funcionamiento | -40 to 85 C | -40 to 85 C |
| Tipos de Antena | Chip Integrated Whip | Chip Integrated Whip |
| Topología | Point to Point, Start | Point to Point, Start, Mesh |

**Tabla [44], Ventajas y desventajas de Xbee**
**Elaborado:** *Autores de la Tesis*

Después de esta breve referencia de dos grandes empresas desarrolladoras y distribuidoras de módulos Zigbee se pudo notar que DIGI al ser un líder en fabricación de estos productos se convierte en la mejor opción a seguir, motivo por el cual se ha escogido los productos de la misma, tomando en cuenta su liderazgo disponibilidad en el mercado, soporte y costo.

En este proyecto, de tesis se empleará dos módulos Zigbee serie 2 debido a que la serie uno no se configura en topología malla.

### 3.14 Análisis de sensores

#### 3.14.1 Sensor de movimiento

Existen varios tipos de sensores capaces de detectar movimiento y como se describió anteriormente se utilizara los sensores infrarrojos pasivos de movimiento y los de microondas cada uno presenta soluciones diferentes y diversas, pero para motivos de mayor seguridad se han optado por los sensores pasivos infrarrojos de movimiento que ofrecen un mayor grado de prevención contra falsas alarmas.

Se utilizará un sensor de  movimiento infrarrojo pasivo con las siguientes características técnicas.

- Voltaje: 12 VDC
- Periodo de calentamiento: 30 segundos.
- Sensibilidad: media
- Altura de montaje: 2.2 m

#### 3.14.2 Detector de Humo

Dos tipos de sensores de humo fueron tomados en cuenta uno es el sensor de uno por defecto fotoeléctrico y el otro es el sensor de humo por ionización características que fueron ya analizadas en el capítulo anterior guiados por los beneficios ofrecidos se optó por utilizar los detectores por ionización por su alto grado de sensibilidad.

### 3.15 Análisis de Microcontroladores

Al hablar y analizar de los microcontroladores se está expuesto a tener un sin número de opciones para escoger, es así que solo con tener claro cuáles son los alcances que tiene el sistema se verá cual es la mejor opción que se adapte y cumpla con los requerimientos necesarios para conseguir un sistema optimo, estable y sin fallas.

En el mercado, se encuentran diferentes marcas de microcontroladores como Motorola, Siemens, Atmel con sus famosos AVRs y uno de los más populares y por ende más usados los PICs de microchip Technology estos últimos combinan una gran calidad, bajo costo y excelente rendimiento, poniendo en consideración la fácil adquisición, costo, desempeño, soporte que ofrecen los microcontroladores PIC se ha considerado en adoptar estos como parte del sistema, a continuación se hará un análisis de cual microcontrolador se utilizará.

#### 3.15.1 Gamas existentes de Microcontroladores

##### 3.15.1.1 Gama baja o básica

Son PICs con una serie de recursos limitados se encuentran con 18 o 28 patitas y se alimentan con una tensión de 2.5v, son ideales para aplicaciones que funcionan con pilas poseen 33 instrucciones con formato de 12 bits, no admite interrupción y la pila tiene 2 niveles.

##### 3.15.1.2 Gama Media

Aquí se encuentra un completo y variado grupo de PICs que abarcan los modelos de 18 a 68 patitas, que controlan varios periféricos.

### 3.15.1.3 Gama Alta

Maneja hasta 58 instrucciones de 16 bits contiene un potente sistema de interrupciones incluye variados controladores de periféricos, puertas de comunicación serie y paralelo, pueden ser aplicados como hardware externo.

### 3.15.1.4 Gama mejorada

La gama mejorada de los PICs fueron diseñadas para soportar aplicaciones avanzadas de automatización cuentan con una gran velocidad (40 Mhz) y un gran rendimiento.

Al hacer una comparación con los Microcontroladores PICs de gama media y alta se pudo notar que existe una marcada diferencia, como es la mayor capacidad de memoria flash, más módulos internos, más pines y su set de instrucciones permite hacer programas en menos espacio. Un punto muy importante de destacar es que entre los PICs 16FXXX y los 18FXXX no existe mucha diferencia de costo y en algunos casos es menor por tal motivo son más cotizados por que al mismo precio se tiene más potencia en los PICs de Gama alta.

Después de analizar los requerimientos para realizar este proyecto de tesis se procede con el diseño del prototipo ahí como el diseño de la interfaz.

# CAPITULO IV

## 4. DISEÑO DEL PROTOTIPO CON FPGA - ZIGBEE

En este capítulo, se presentará el diseño en hardware iniciando desde la elaboración del esquemático hasta culminar con la elaboración de la placa del prototipo para comunicación FPGA - ZIGBEE - PC. Así mismo, se describirá el diseño del software tanto para el microcontrolador como los componentes ADC y UART para el FPGA, finalmente se desarrollará una interfaz en la computadora para la recepción de datos.

## 4.1 DISEÑO DE HARDWARE

Una vez estudiado los conceptos básicos sobre los componentes electrónicos del prototipo y como estos componentes mediante módulos se comunican por medio de la red FPGA - ZIGBEE - PC, Así como también, se estudió a los microcontroladores y el entrenador para FPGAs SPARTAN 3E y particularmente las características del PIC 16F876 que por su puesto tiene integrado un módulo de comunicación UART. Con estos conocimientos, se procederá a la elaboración del Hardware, para este fin, se utilizará el software de creación de esquemáticos y simulación PROTEUS 7.7 SP2 modulo ISIS y para el FPGA se utilizará el lenguaje VHL en el aplicativo ISE 11.1de Xilinx, descrito en detalle en el capítulo 3.

## 4.2 ELABORACIÓN DEL ESQUEMÁTICO

Para la elaboración del prototipo de comunicación FPGA - ZIGBEE - PC, se utilizarán dos módulos, el primero llamado **Adquisición**, este módulo está constituido por el entrenador Sparta 3E, donde se diseñará un componente para ADC[29], al cual se le

[29] Conversión analógica-digital consiste en la transcripción de señales analógicas en señales digitales

conectará un sensor de temperatura LM35D, de humo y un sensor de movimiento, la señal del ADC, será enviada a un componente UART y la señal de TX será enviada al segundo módulo de Comunicación RF que consta de un transmisor y receptor ZIGBEE, la señal del receptor será recogida y reenviada a un dispositivo llamado EXPLORER, esta enviará la señal serial a través de un cable USB a la PC, cabe mencionar que la transmisión no es con tecnología USB, sino serial, es decir, vía RS232[30], también tendrá algunos elementos adicionales analizados y seleccionados en el capítulo 3.

Para complementar todo el sistema de adquisición y comunicación, también se requieren elementos electrónicos adicionales como se muestra en el diagrama de bloques en la siguiente figura.

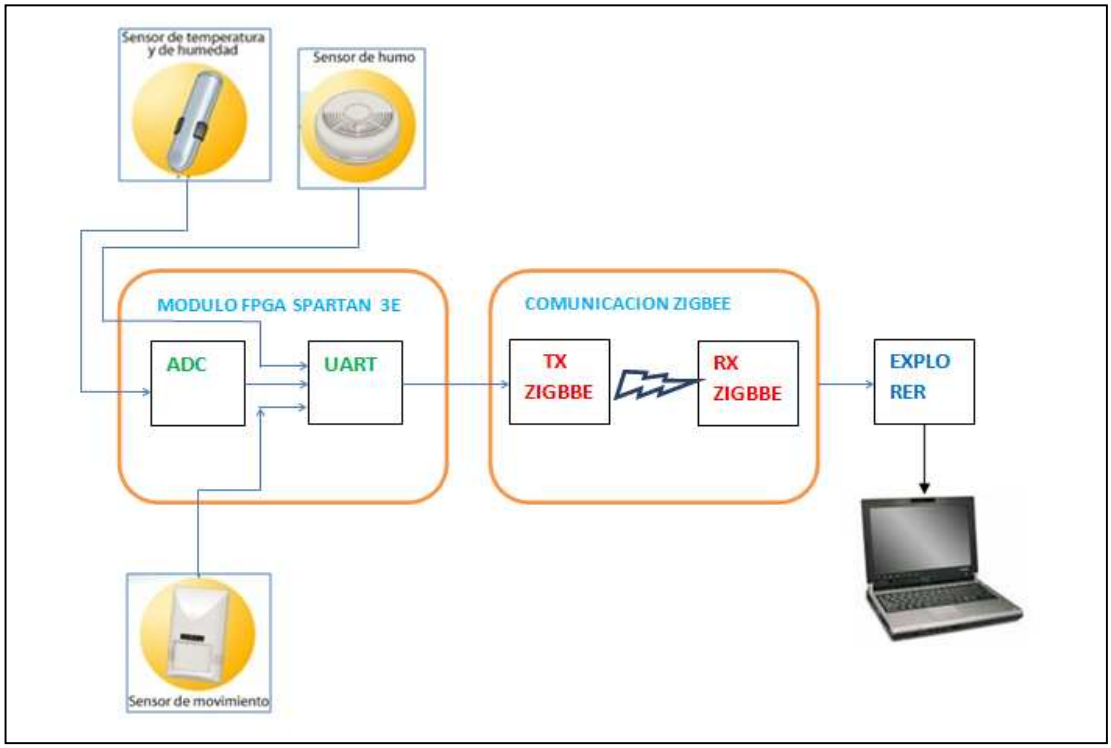



**Figura [14], Diagrama de bloques para la comunicación**

**Elaborado:** *Autores de la Tesis*

[30] Es una interfaz que designa una norma para el intercambio serie de datos binarios

En el diagrama de bloques anterior, se observa un panorama general del proyecto y la conexión entre módulos como los sensores, la tarjeta entrenadora para FPGA Spartan 3E y la comunicación inalámbrica con módulos zibgee hacia la PC, como se muestra en la siguiente figura.

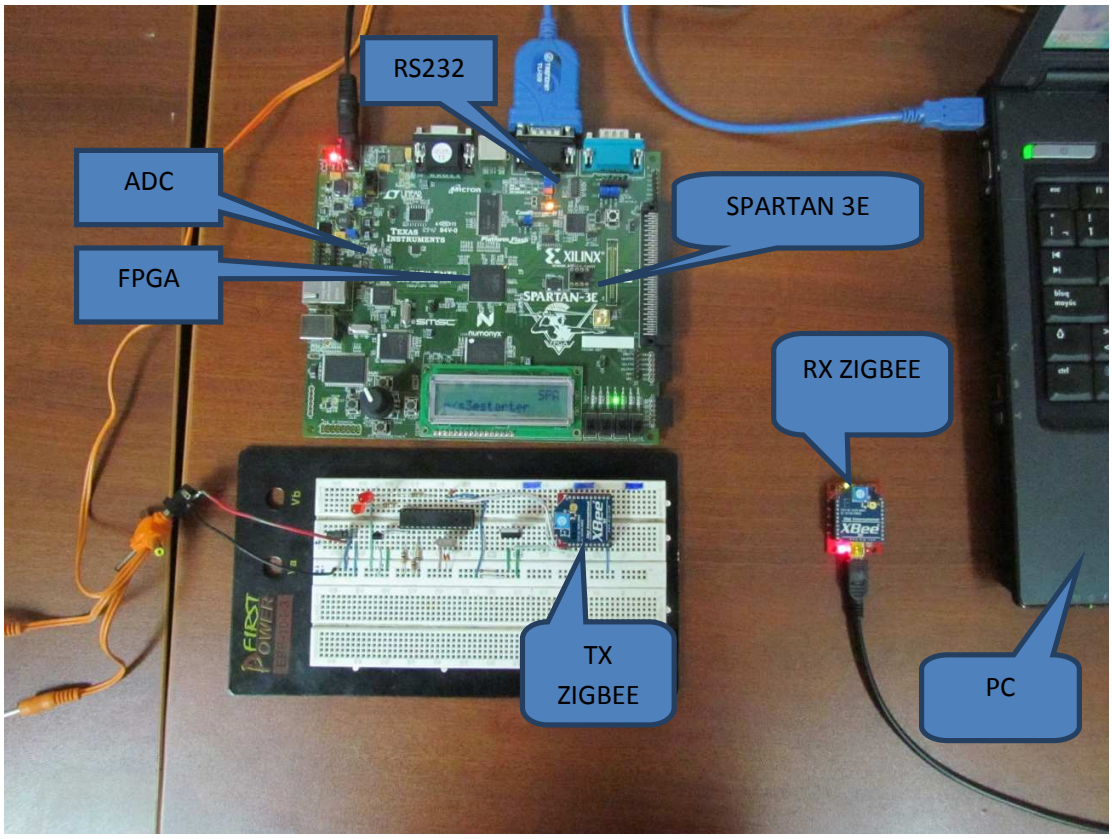



**Figura [15]**, **Comunicación Entrenador FPGA – PC**

**Elaborado:** *Autores de la tesis*

En la figura anterior del proyecto, los sensores son conectados, el de temperatura al módulo ADC y los otros a las entradas digitales, estos datos son enviados a través del UART a la PC.

Adicional al FPGA, se desarrolló una conexión con microcontrolador para probar la transmisión de datos por Zigbee, puesto que el módulo de transmisión RS232 del

entrenador entrega directamente las señales al conector DB9[31] y no fue posible realizar una conexión intermedia en la tarjeta porque se dañaría la misma.

## 4.3 DESCRIPCION DE LOS MODULOS DE COMUNICACIÓN FPGA - PC

### 4.3.1 MODULO ADQUISICION

En el capítulo 3, también se analizaron los elementos de Hardware principales que podrían servir para este proyecto. Por tanto, en la fase de diseño se procedió a utilizar aquellos componentes con mejores características y favorables al proyecto. Es así que, para el modulo de adquisición, se utilizará el entrenador Spartan 3E que está equipado con un conversor ADC de 12 bits, un preamplificador y un control SPI. A la entrada del ADC está conectado a un sensor semiconductor de temperatura tipo LM35D.

El sensor puede medir la temperatura en un rangode 0C a 100C y genera un voltaje analógico directamente proporcional a la medición de temperatura (es decir, la salida es 10mV/C). Por ejemplo, a 20C la salida de voltaje será de 200mV. Ver siguiente figura.

[31] Es un conector analógico de 9 clavijas de la familia de conectores D-Subminiature

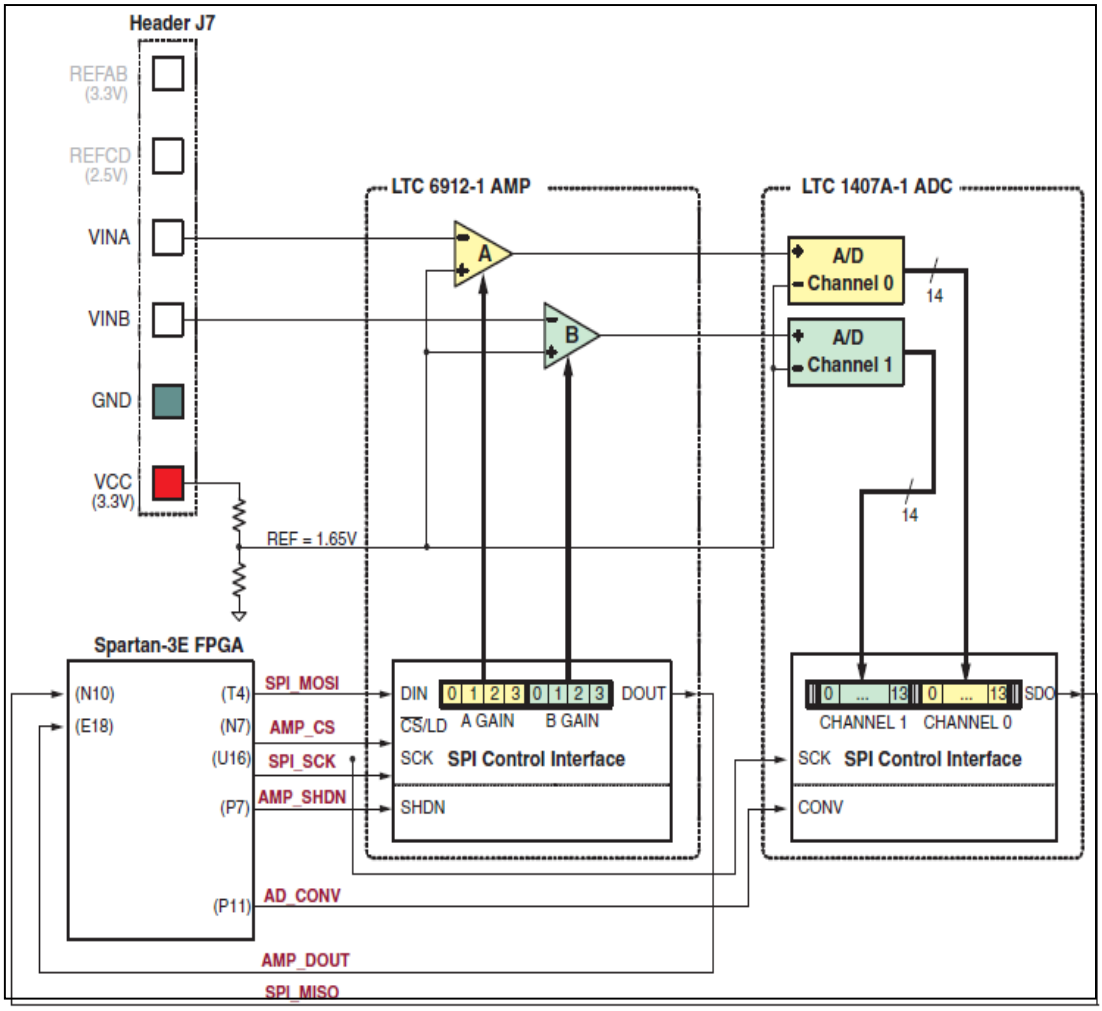



**Figura [16], Conversor ADC de 12 bits**

**Tomado Manual Xilinx**

### 4.3.2 MODULO UART

El proceso de recepción y envío de datos, se forma de un componente UART diseñado y grabado en el entrenador, el cual recibe los datos del ADC y envía estos al transmisor Zigbee. Ver siguiente figura.

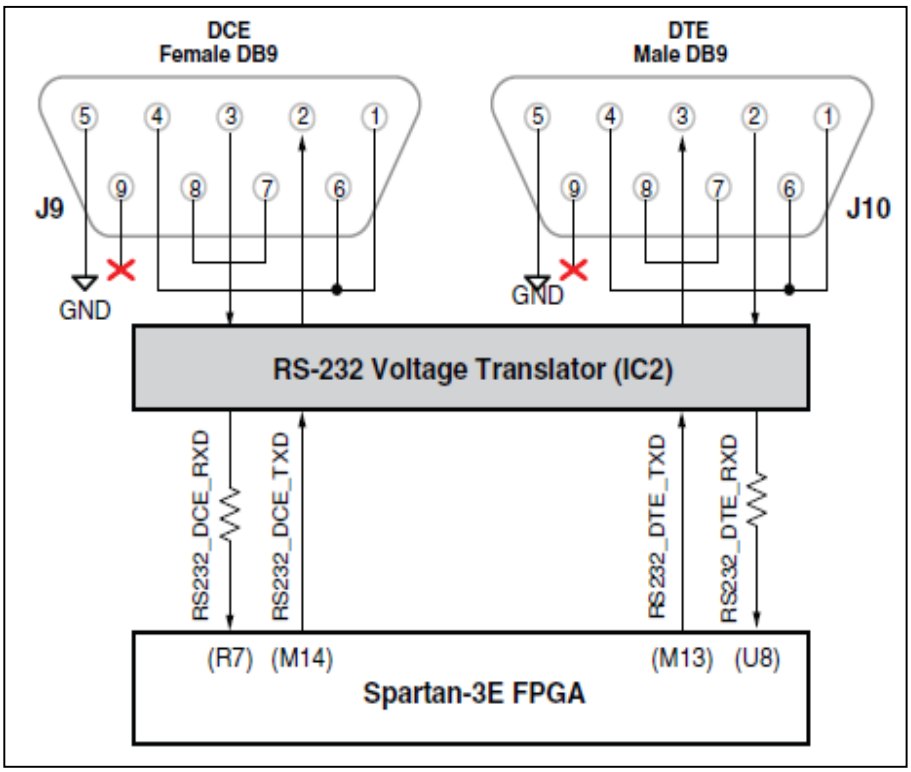



**Figura [17], Proceso de recepción y envío de datos**

**Tomado Manual Xilinx**

### 4.3.3 MODULOS ZIGBEE

También, en el capítulo 3, se detalló este modulo electrónico de comunicación serial, como se muestra en la figura, este módulo, se conecta  a los pines RC6 y RC7 del PIC, es decir, al modulo UART integrado en el mismo.Tanto las línea de transmisión como recepción son conectados a los pines respectivos del módulo Zigbee, como se muestran en las siguientes figuras.

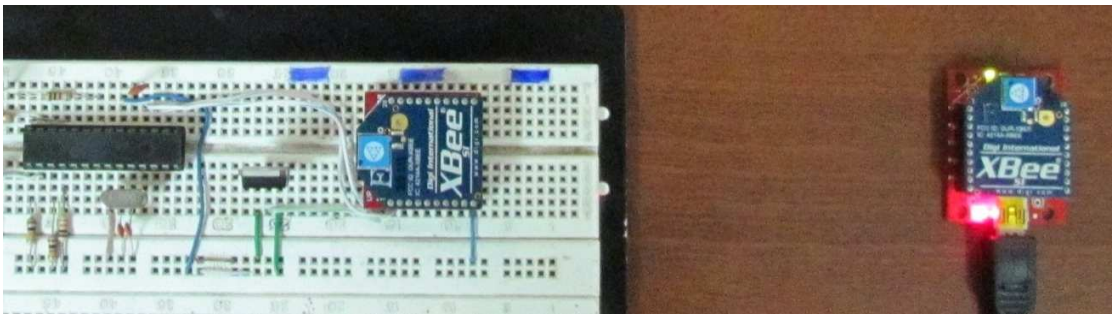



**Figura [18], Pines de Comunicación Zigbee**

**Elaborado:** *Autores de la tesis*


**Figura [19], Módulo y dispositivo Zigbee**

**Elaborado:** *Autores de la tesis*

## 4.4 SIMULACION CON MICROCONTROLADOR

Para el proceso de simulación de comunicación FPGA – ZIGBEE - PC, se requirió del modulo ISIS de PROTEUS, puesto que una de las ventajas de este aplicativo es su simulador para microcontroladores avanzado. Además este simulador viene equipado con el sensor de temperatura LM35, el micro utilizado fue el 16F873A, pulsadores y un terminal virtual para transmisión serial con RS232. Tal vez como desventaja se podría citar que hasta la versión actual no posee las librerías para simular FPGAs todavía, por

tal razón se utilizó en vez del FPGA un microcontrolador el cual tiene como entradas el sensor de temperatura y pulsadores para simular los otros sensores como el de humo y movimiento y la salida va directamente conectado al terminal virtual.

En la siguiente figura, se muestra dicha simulación.

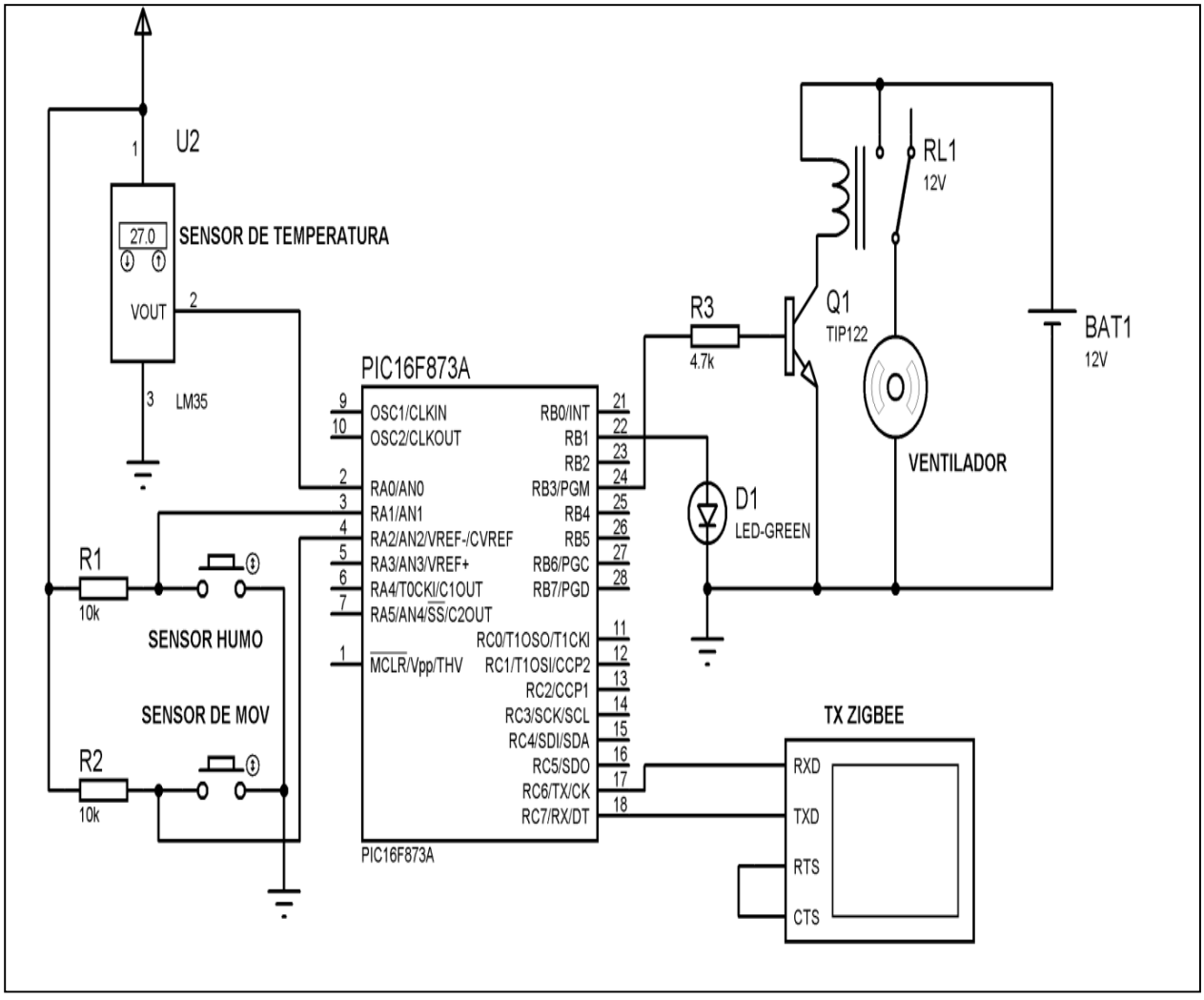



**Figura [20], Esquemático Comunicación ZIGBEE**

**Elaborado*:*** *Autores de la tesis*

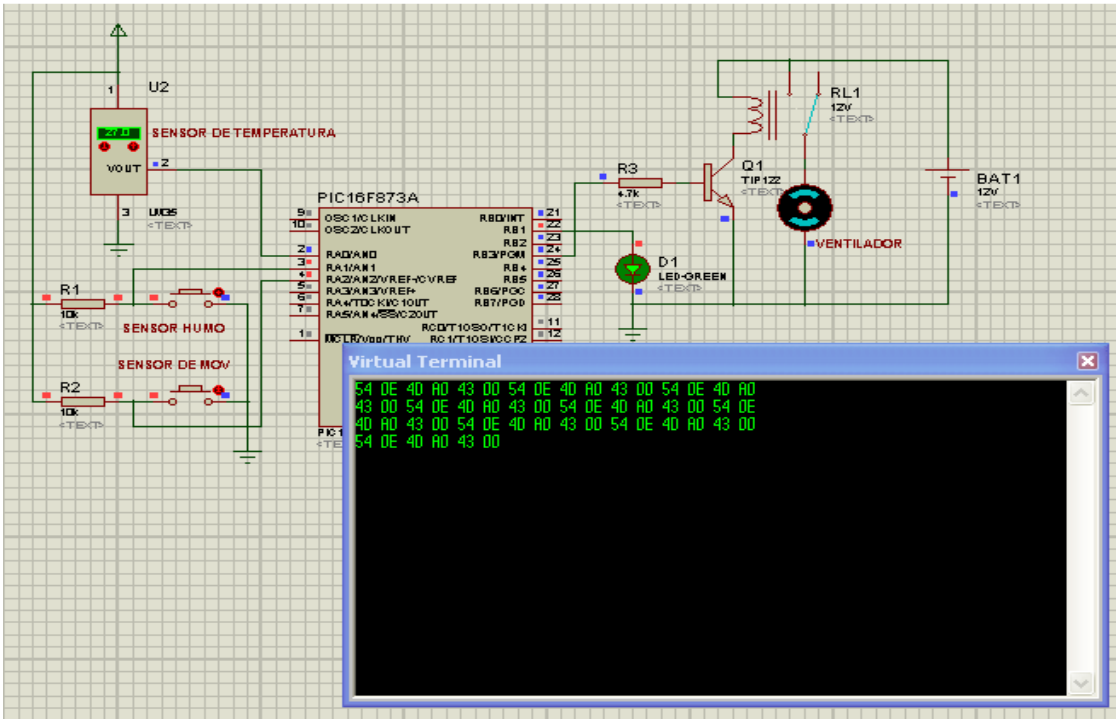



**Figura. [21] ,Simulación Comunicación ZIGBEE**

**Elaborado:** *Autores de la tesis*

Figura 21 se puede observar en la ventana virtual la llegada de datos, asi como el central hacia los actuadores.

## 4.5 DISEÑO CON MICROCONTROLADOR DEL PROTOTIPO PARA CONEXIÓN CON ZIGBEE

Una vez realizado el diseño de comunicación FPGA - ZIGBEE y posteriormente simulado, se procedió a implementarse en el protoboard, el mismo que está compuesto por un sensor de temperatura, entradas para los otros sensores digitales, un microcontrolador 16F873A, reguladores de voltaje, el módulo ZIGBEE transmisor, adicionalmente se añadirán un módulo de potencia para control del ventilador.

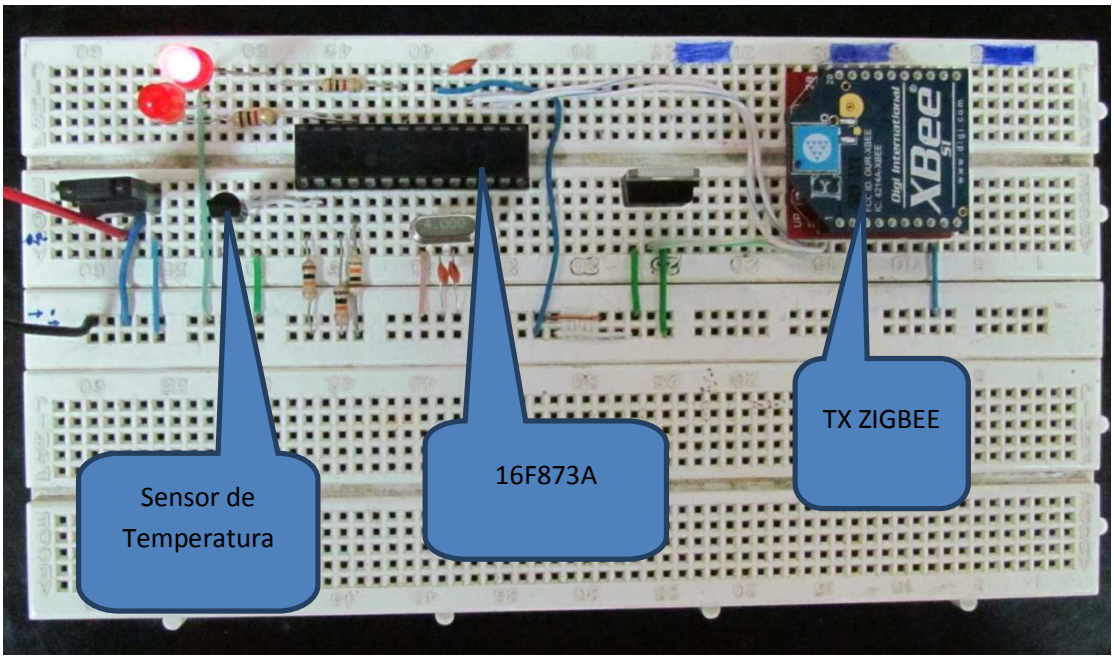



**Figura [22], Implementación en proto de la comunicación FPGA – ZIGBEE**
**Elaborado:** *Autores de la tesis*

Debido a que es solo un circuito de prueba, no se precederá a diseñar la placa, además el entrenador Spartan 3E viene con todas las conexiones listas.

## 4.6 DISEÑO DEL SOFTWARE

El diseño del software para este proyecto, se divide en tres módulos generales: Diseño del programa para el Microcontrolador, para adquisición, transmisión ZIGBEE y RS232 hacia la computadora, diseño del programa VHDL jerárquico para el FPGA y la Interface en la PC, la misma que recibe los datos por el puerto serial y procesa de acuerdo a los requerimientos dados por el usuario.

## 4.7 DISEÑO DEL PROGRAMA PARA EL MICROCONTROLADOR

Para el diseño del programa para el micro, se utilizó la aplicación MIKROC PRO V 4.15 de la empresa MIKROELEKTRONIKA por las razones citadas en el capítulo 3.

En este proyecto la tasa de transmisión es de 9.6 kb/s. Con una frecuencia de reloj de 4MHz cuando se utiliza microcontrolador, mientras que, para la transmisión por el UART del FPGA es de 19200 bps.

A continuación, se muestran el programa, pero antes de detallar el listado del programa, primero se muestra dicho diseño con lenguaje descriptivo, luego el diagrama de flujo.

**NODO ADQUISICION**

**Lenguaje Descriptivo**

Inicializar móduloUART

Configurar parámetros iniciales

**HACER SIEMPRE**

Espera un dato el receptor del UART para iniciar la TX

Leer el valor de la temperatura por el canal 0

**SI** temperatura es mayor a Tmax”

Activa ventilador

**FIN SI**

**SI** sensor_humo es activado

Envía carácter H a la PC

**FIN SI**

**SI** sensor_mov es activado

Envía carácter V a la PC

**FIN SI**

Envía carácter T de cabecera y el valor de temperatura

**FIN HACER**

### 4.7.1 Diagrama de Flujo del microcontrolador

En la siguiente figura, se muestra el diagrama de flujo para la transmisión y recepción de datos.

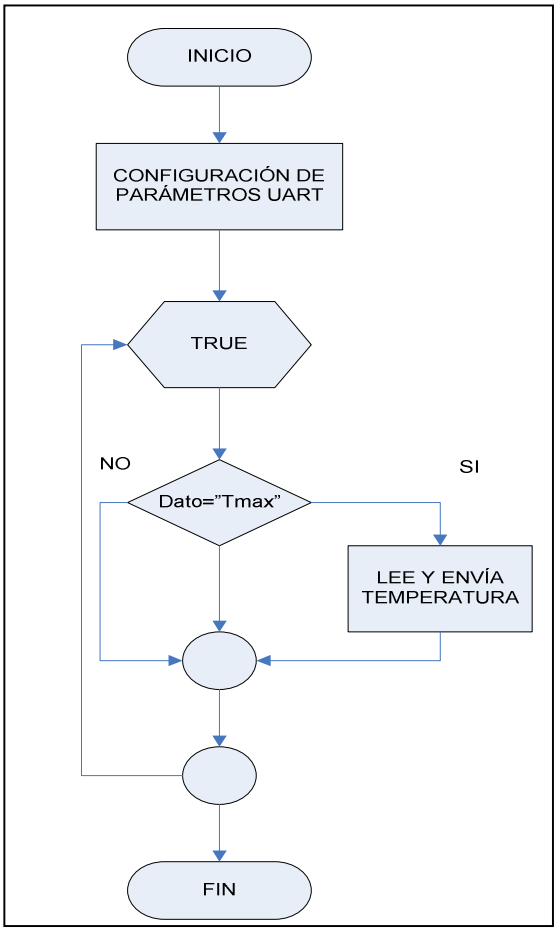



**Figura [23], Diagrama de flujo TX y RX**

**Elaborado:** *Autores de la tesis*

## 4.8 PROGRAMA MICROCONTROLADOR

En el siguiente listado, se muestra el programa para la transmisión y recepción de los datos para el micro.

```
// Programa para la Tx – Rx para micro 16F873A
// Realizado por: Castillo Darwin y Osorio Richard
// UPS – Quito
//
#define led_func_tx portb.f1
#define temperatura_critica portb.f2
#define rele_ventilador portb.f3
#define sensor_humo porta.f1
#define sensor_mov porta.f2

unsigned short temperatura,Tmax,dato,bandera,caracter;

voidinterrupt()    // Interrup para la Rx de datos desde la PC
{
bandera = 0;

if (PIE1.RCIE)
  {
dato = UART1_Read();
Eeprom_Write(0x1, Tmax);
```

```
if (dato=='A')
   {
caracter='A';
bandera = 0;
   }
if (dato=='E')
   {
caracter='E';
bandera = 1;
   }
if (dato=='I')
   {
Tmax = Eeprom_Read(1);
Tmax = Tmax + 5;
if (Tmax>=250)
      {
Tmax=250;
      }
Eeprom_Write(0x1, Tmax);
   }
if (dato=='D')
   {
Tmax = Eeprom_Read(1);
Tmax = Tmax - 5;
if (Tmax<=5)
```

```
      {
Tmax=5;
}
Eeprom_Write(0x1, Tmax);
   }
}
}

voidmain() {
UART1_Init(9600);       // Velocidad de Tx
  adcon1 = 0x8E;          // Configuración de AN0 analógica y el resto digitales
trisa  = 0xFF;          // Puerto A como entradas
trisb = 0;              // Puerto B como salidas
portb = 0;              // Puerto B = 0
  INTCON = 0xC0;          // Habilitado Interrupciones globales y perifericas
bandera = 0;
Tmax = 160;

do
  {
if (UART1_Data_Ready())  // Si existe un dato en el Buffer de Rx, entonces
   {                     // habilita el bit de interrupción en la RX (RCIE)
PIE1 = 0x20;
   }
```

```
led_func_tx = 1;
delay_ms(200);
led_func_tx = 0;
delay_ms(200);

temperatura = ADC_Read(0) >> 2;  // Lee el valor de humedad digitalizado

if (temperatura >= Tmax || bandera == 1)
   {
temperatura_critica = 1;
rele_ventilador = 1;
   }
else
   {
temperatura_critica = 0;
rele_ventilador = 0;
}
//-----------------------------
if (sensor_humo == 0)
   {
delay_ms(30);
UART1_Write('H');              // Enviacodigo para sensor de humo activado
}

if (sensor_mov == 0)
```

```
   {
delay_ms(30);
UART1_Write('V');              // Envía código para sensor de mov activado
   }


   UART1_Write('T');           // Envía una cabecera para el valor de humedad
   UART1_Write(temperatura);   // Envía el valor de la temperatura

UART1_Write('M');
   UART1_Write(Tmax);
   UART1_Write('C');
   UART1_Write(caracter);
} while (1);                   // Fin de ciclo
}
```

## 4.9 DISEÑO DEL PROGRAMA PARA EL FPGA

Para el FPGA, se utilizó lenguaje VHDL en todos sus componentes, así, se necesitó de componentes principales como el ADC y el UART, a su vez, estos, se dividieron en sub componentes para que la programación sea un poco sencilla, es decir, se utilizó diseño jerárquico con los componentes.

El proyecto principal de denominó ***adc-uart,*** a continuación, se crearon componentes por separado como son el ***adc y uart*** por otro lado, ahora bajo el uart, se tienen los componentes para la transmisión denominada ***uart-transmit*** y ***uart-recept***, además de

estos componentes, se utilizaron también un generador de baudios, un componente de desplazamiento ***buffer fifo.*** Finalmente se empaquetan estos componentes y se llaman desde el ***proyectoadc-uartque*** es el nivel principal. Ver siguiente figura.

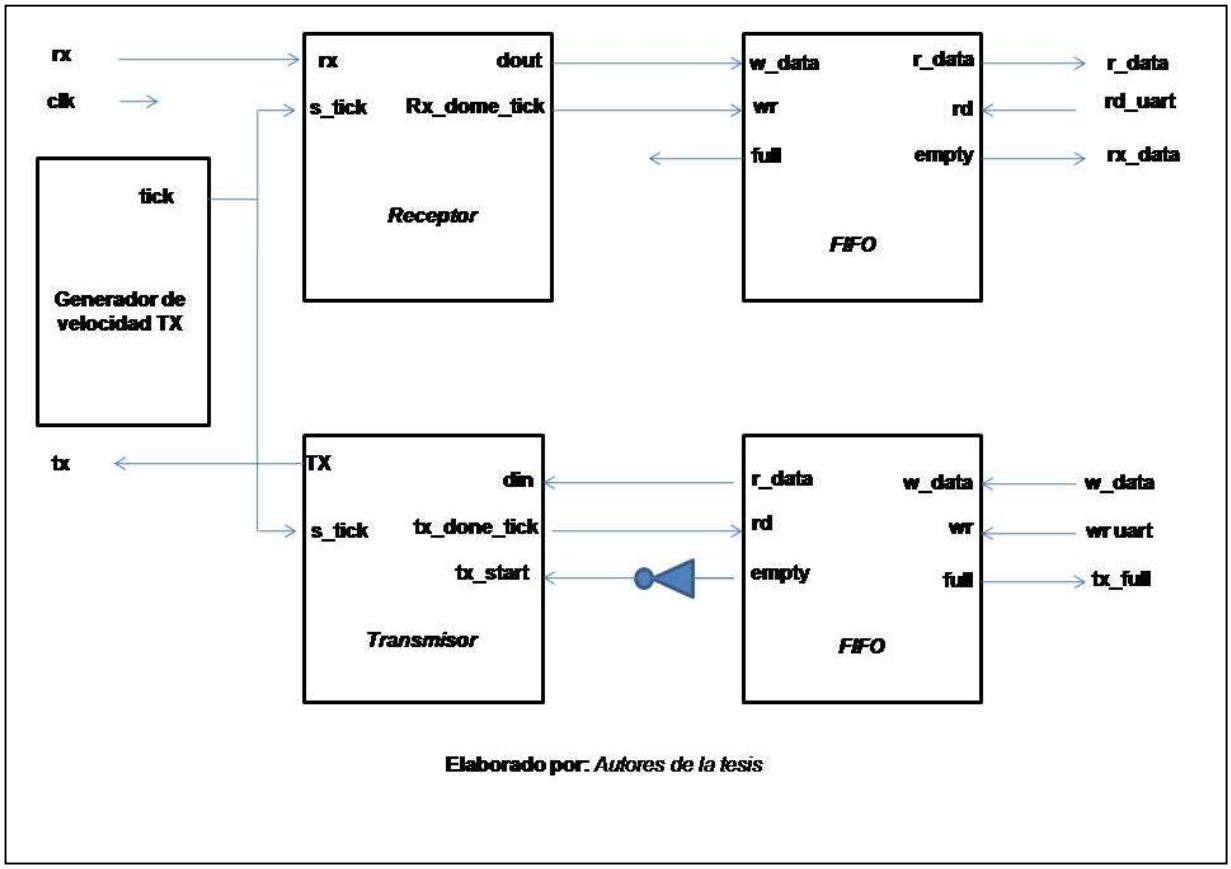



**Figura [24], Componentes del UART**
**Elaborado:** *Autores de la tesis*

Una vez editado cada uno de los listados en el ISE de XILINX, se procede a empaquetar, es decir, a unir todos los componentes independientes para obtener un solo proyecto. Después de varios intentos, finalmente no se tuvieron cero errores, entonces,se procedió a asignar pines físicos del FPGA con sus respectivas interfaces, es decir, se generó un archivo UCF. Por último, se compiló todo el programa para obtener un archivo adc-uart.bit necesario para grabar en el FPGA para posteriormente realizar la transmisión de los datos hacia la PC.

## 4.10 DISEÑO DE LA INTERFAZ PARA LA PC

En la siguiente pantalla se muestra la interfaz grafica donde se puede observar los tres componentes o sensores del prototipo.

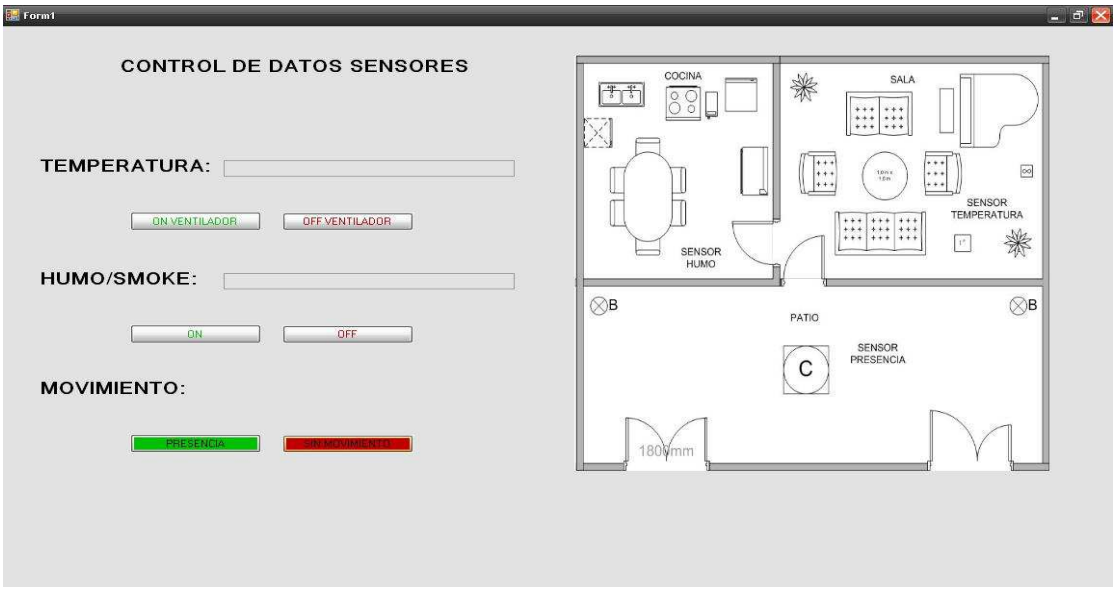



**Figura [25], Interfaz general del prototipo en la PC**
**Elaborado:** *Autores de la Tesis*

En la siguiente pantalla se muestra la ubicación del sensor de temperatura el cual será monitoreado por la interfaz gráfica.

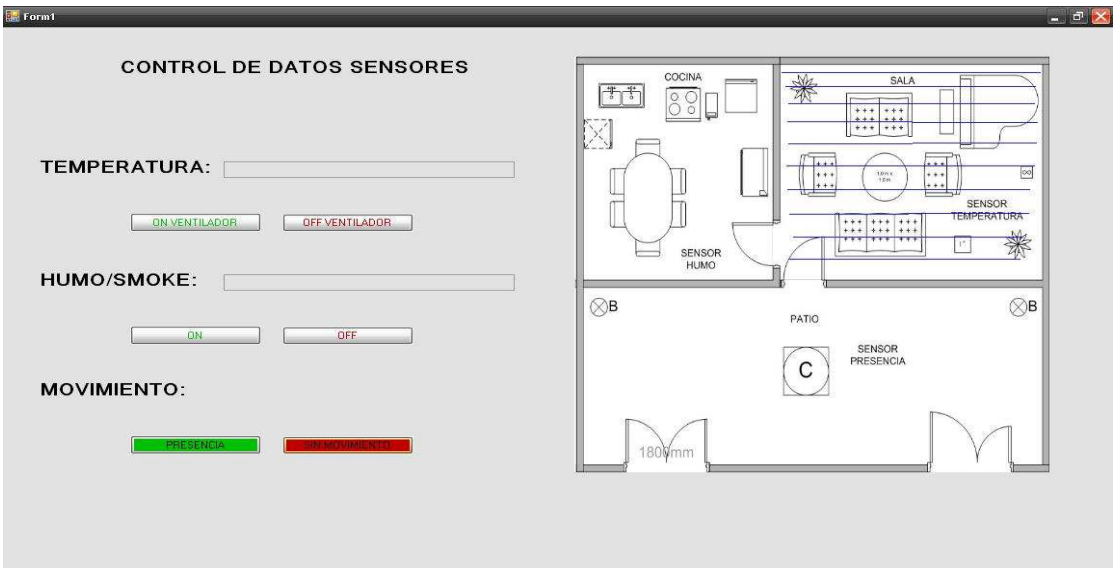



**Figura [26], Interfaz de control para temperatura**
**Elaborado:** *Autores de la Tesis*

En la siguiente pantalla se muestra la ubicación del sensor de movimiento el cual será monitoreado por la interfaz gráfica.

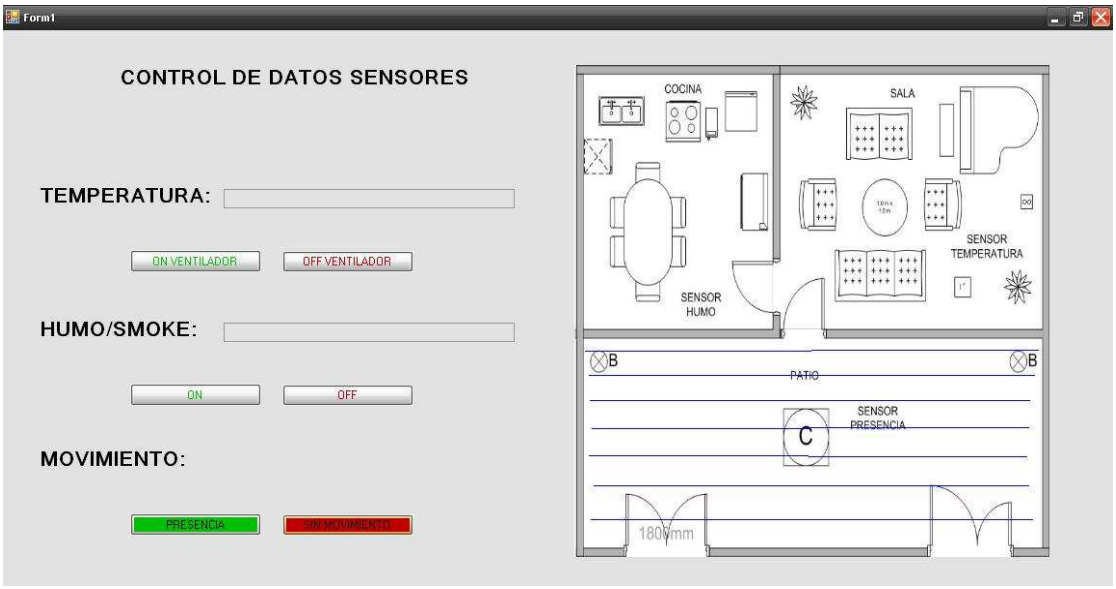



**Figura [27], Interfaz de control para sensor de movimiento**
**Elaborado:** *Autores de la Tesis*

En la siguiente pantalla se muestra la ubicación del sensor de humo el cual será monitoreado por la interfaz gráfica.

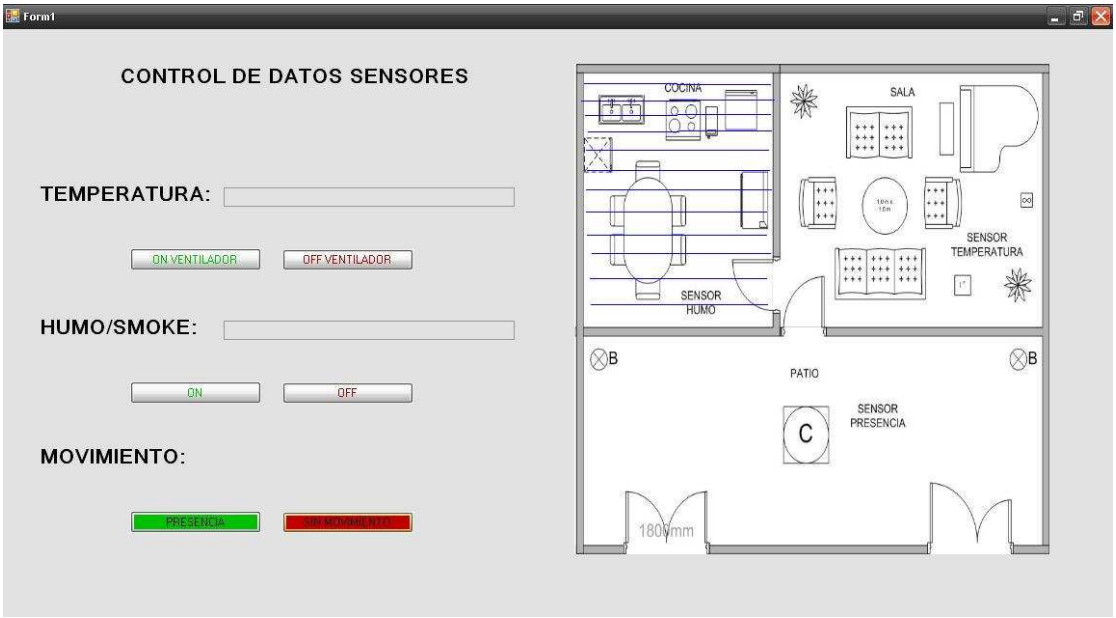



**Figura [28], Interfaz de control para sensor de humo**
**Elaborado:** *Autores de la Tesis*

## 4.11 DESCRIPCION DE LA INTERFAZ

Para el desarrollo de la interfaz gráfica se utilizo el IDE Sharp Developer por todo lo descrito en el capítulo III. La interfaz está compuesta por tres ventanas o módulos de supervisión que abarcan los espacios dentro del área de la domótica, teniendo:

Sala: Donde se encuentra el sensor de temperatura

Cocina: Donde se encuentra el sensor de humo

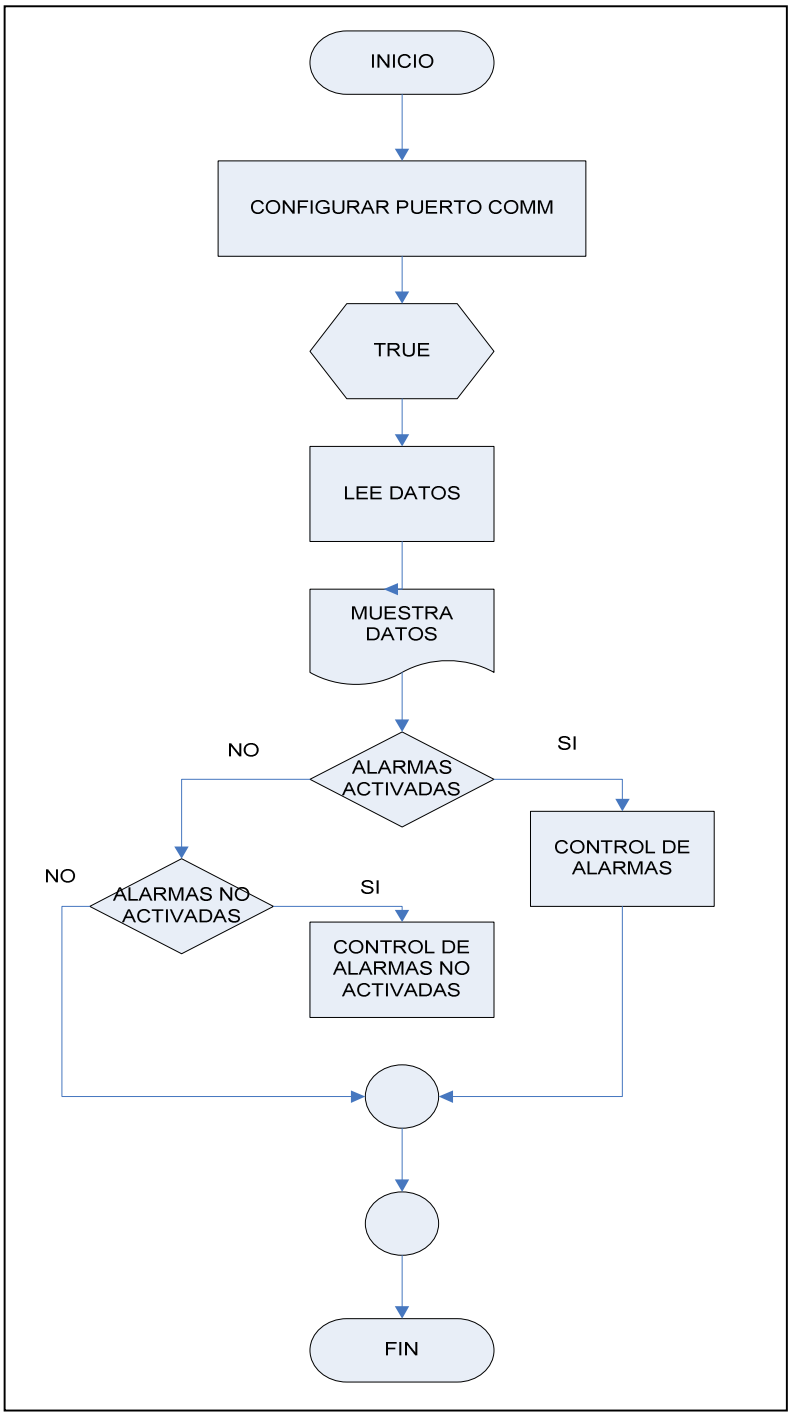
Patio: Donde se encuentra el sensor de movimiento.

Se tiene un control de los tres sensores los cuales están enviando información hacia la Interfaz realizada en la PC. Al momento de activarse un sensor se activa una alarma hacia la interfaz y se puede visualizar la misma en forma grafica, y la misma se puede desactivar.

En el siguiente diagrama de flujo se muestra las diferentes etapas que cumple la interfaz gráfica dentro del prototipo. Figura 29

**Figura [29]**

**Diagrama de flujo para la interfaz**




**Elaborado:** *Autores de la tesis*

El listado del programa en para adquirir datos de los sensores, se muestra en el **anexo 5**

## 4.12 PRUEBAS Y RESULTADOS

En este apartado, se describe las diferentes pruebas que se realizaron al prototipo, así como los resultados que se obtuvieron fruto de los varios laboratorios que se hicieron a lo largo del periodo de realización de la tesis.

### 4.12.1 Pruebas del Prototipo

Una vez que se realizó el diseño de transmisión de datos mediante la tecnología Zigbee, se procedió a simular en el programa ISIS de PROTEUS, en primera instancia, las pruebas de simulación se realizaron con el circuito, el microcontrolador y luego con el FPGA, se recibían datos a la PC a través de comunicación serial utilizando protocolo RS232.

Obteniéndose, resultados exitosos en simulación, se procedió a implementar este diseño en un protoboard, en el cual estaba armado tanto el módulo de adquisición como el nodo módulo de transmisión.

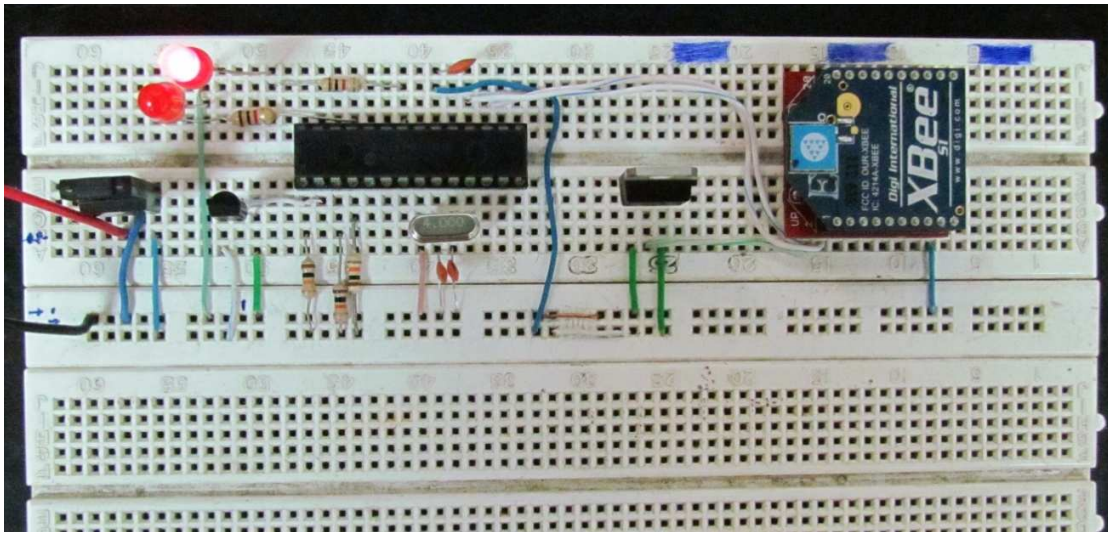



**Figura [30], Implementación de los nodos Adquisición y Principal en protoboard.**
**Elaborado:** *Autores de la tesis*

A continuación, se muestra el prototipo completo e interconectado a la computadora y transfiriendo los datos. Se puede observar que tanto el nodo de adquisición como el nodo principal tienen un sistema de señalización que consta de un led bicolor, el cual titila en color rojo cuando entre en funcionamiento los módulos de comunicación. Así mismo, el medio de transmisión de la tecnología Zigbee posee dos leds para RX y TX de datos.

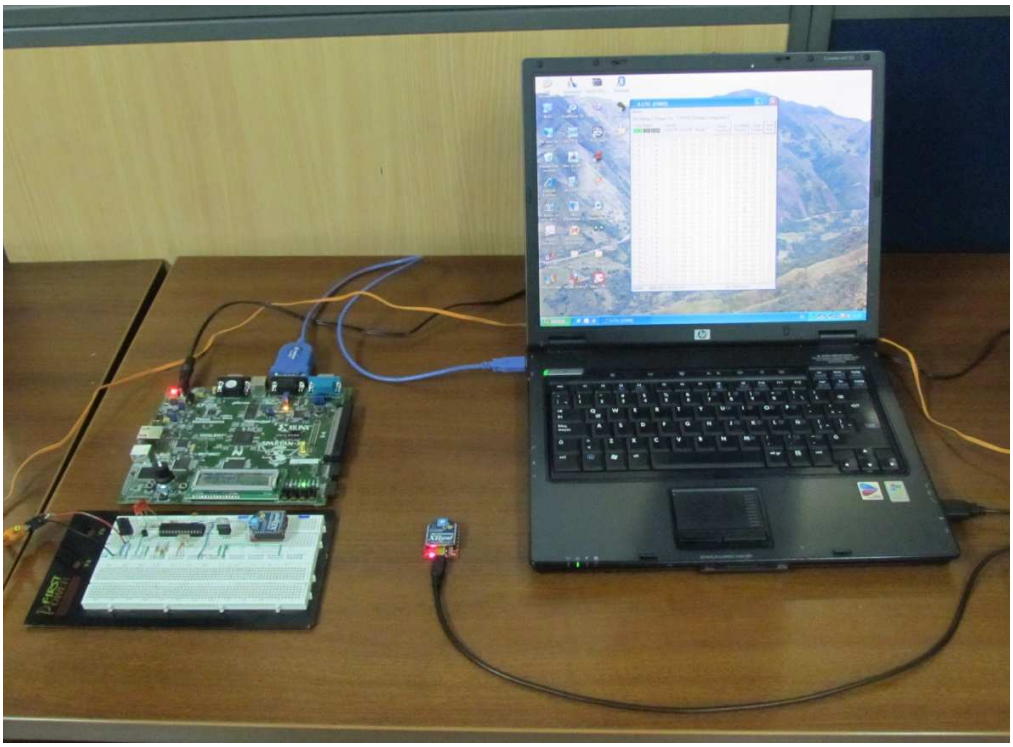


**Figura [31], Prototipo completo entre Proto – Zigbee - PC**
**Elaborado:** *Autores de la tesis*

A continuación se muestra los datos obtenidos mediante el programa X-CTU el cual el mismo que puede ser configurado de acuerdo a nuestras necesidades de visualización.

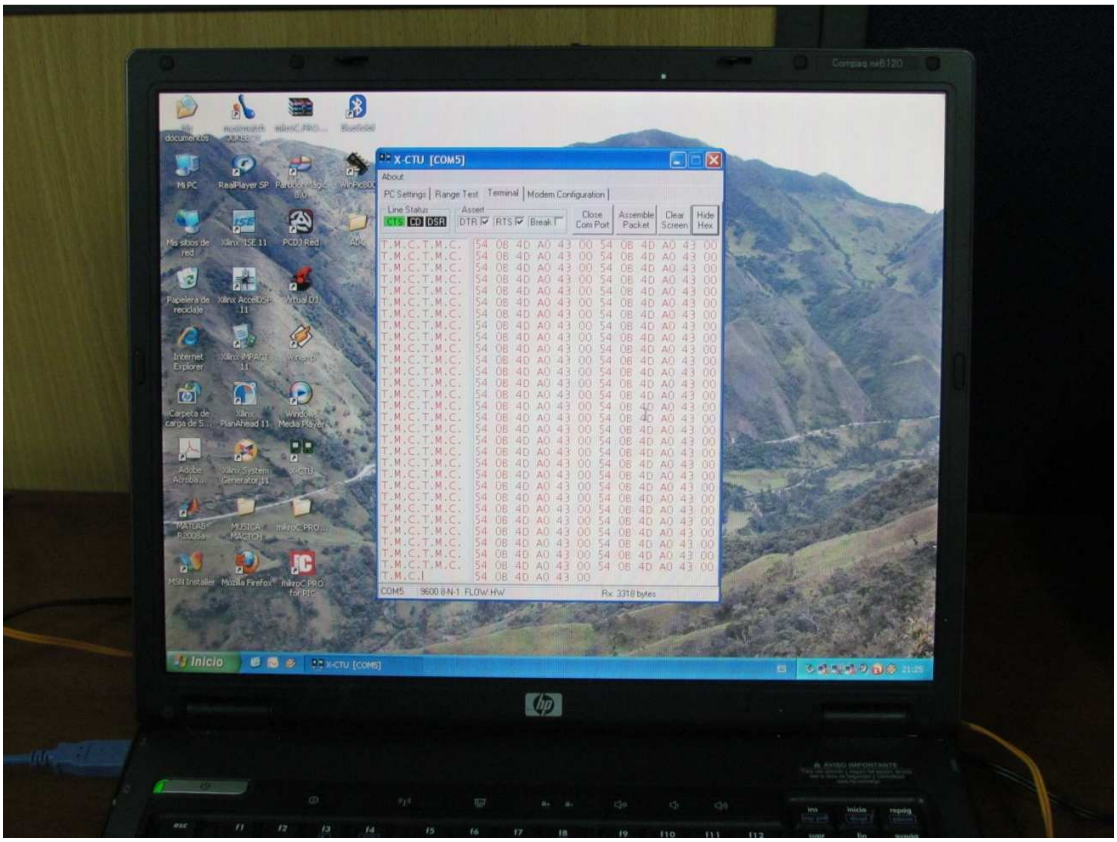



**Figura [32], Datos obtenidos por el receptor X-CTU**

**Elaborado:** *Autores de la tesis*

Después de realizar las pruebas de recepción con el programa X-CTU procedemos a probar con el programa X-CTU el mismo que fue elegido por sus ventajas en el capítulo III, igualmente este programa nos muestra una interfaz en la cual podemos observar los datos en forma de binarios o hexadecimales.

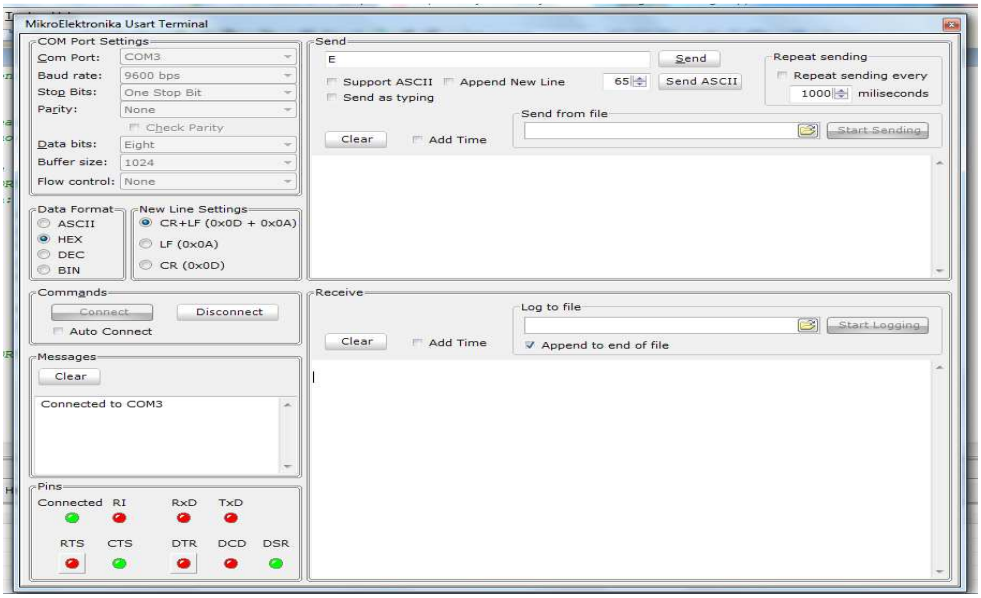



**Figura [33], Datos obtenidos por el receptor MiKroC Pro**

**Elaborado:** *Autores de la tesis*

En la siguiente figura se muestra la recepción de datos mediante el entrenador de FPGA, se puede observar que el entrenador se encuentra enviando datos del sensor de calor hacia la interfaz de la PC.

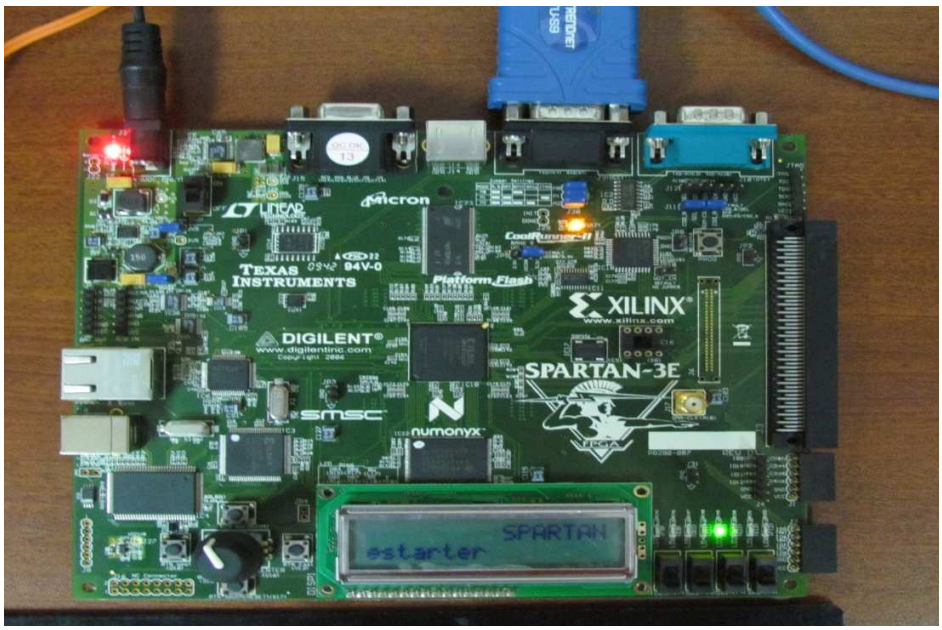



**Figura [34], Datos obtenidos por el receptor MiKroC Pro**

**Elaborado:** *Autores de la tesis*

# CONCLUSIONES Y RECOMENDACIONES

## CONCLUSIONES

En la investigación realizada, se pudo constatar que no existen estudios, ni prototipos del entrenador FPGA Spartan 3E utilizando tecnología inalámbrica, siendo este un factor negativo para la implementación del prototipo.

Sin un conocimiento previo sobre el entrenador FPGA 3E se dificulta la elaboración de prototipos ya que esto implica realizarlo desde cero.

Con un estudio más amplio de esta nueva tecnología se puede realizar aplicaciones de gran escala dentro del área de la domótica. Así mismo son tarjetas inteligentes que se pueden reprogramar y reutilizar de acuerdo a las necesidades de los desarrolladores.

El desarrollo de una red mediante dispositivos que cuenten con tecnología Zigbee hoy en día es una manera eficiente, económica y sobre todo practica de construir un sistema domótico que abarque tres puntos básicos: seguridad, disponibilidad y convergencia.

La forma de transmisión utilizando tecnología Zigbee, simplifica enormemente la comunicación entre módulos electrónicos debido al ahorro de material en el cableado, lo que hace que los dispositivos de comunicación sean portables para velocidades de transmisión en tiempo real.

En el prototipo que se desarrollo, se aprecio que la tecnología Zigbee puede transmitir señales al receptor en la posición en la que este encuentre dentro de una red de área personal.

# RECOMENDACCIONES

Realizar un estudio de mercado para identificar la demanda de estos sistemas de comunicación inalámbrico basados en el entrenador FPGA 3E orientados para las seguridades dentro de la domótica.

Para el desarrollo de futuros proyectos es de vital importancia tener una relación directa con el proveedor del entrenador FPGA Spartan 3E para un conocimiento técnico del producto y los tiempos de entrega en el país.

Al trabajar con el entrenador FPGA 3E de debe tener en cuenta que los dispositivos incorporados en el mismo no se les puede desprender ni manipular físicamente ya que esto puedo causar daños irreversibles dentro del entrenador.

Se debe dar un seguimiento profundo al lenguaje de programación VHDL ya que el mismo controla mediante programación los puertos I/O dentro del entrenador FPGA 3E como es UART.

En cuanto al sensor de temperatura que se utilizo en este proyecto, se recomienda para futuros proyectos utilizar uno de mayor precisión y de conmutación rápida.

Al escoger el dispositivo de comunicación inalámbrica Zigbee se debe tener en cuenta que el dispositivo a escoger sea compatible con las demás tecnologías ya existentes.

# BILBIOGRAFÍA

## TEXTOS


1. **CORRALES V. SANTIAGO,** ” Electrónica Práctica con Microcontroladores PIC” Agosto 2006,Ecuador

2. **TOJERO GALARZA GERMAN,** “Proteus simulación de circuitos electrónicos y microcontroladores atravez de ejemplos “Primera edición , Grafica Diaz ,2009”.

3. **REYES A. CARLOS,** “Microcontroladores PIC Programación en Basic ”, Segunda edision, Quito-Ecuador 2006.

4. **GARCIA BREIJO EDUARDO**” Compilador C CSS y simulador de PROTEUS para Microcontroladores PIC” , Segunda Edición, Editorial Marcombo Barcelona-España 2009


## PAGINAS WEB


**1. REDES ZIGBEE, 2011,**

http://www.blogelectronica.com/redes-zigbee-i-introduccion/

**2. MICROCHIP, 2008,**

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2551.

**3. ZIGBEE ALLIANCE, 2002,**

http://www.zigbee.org.

**4. SEGURIDADA INFORMATICA, 2008,**

http://foro.elhacker.net/printpage.html;topic=161120.0.

**5. DIGI,AGOSTO 2011,**

http://www.digi.com/products/usb/anywhereusb.jsp.

**5. FPGA 2010,**

http://es.wikipedia.org/wiki/FPGA

**6. DEMODESK, 2000**

http://www.domodesk.com/content.aspx?co=97&t=21&c=47

**7. SUPPORT SHARP, 2010**

http://sourceforge.net/projects/sharpdevelop/support

**8. NEOTEO, 2005**

http://sourceforge.net/projects/sharpdevelop/support

**9. ESPACIO LINUX, 2007**

http://www.google.com.ec/url?sa=t&source=web&cd=4&sqi=2&ved=0CCcQFjAD&url=http%3A%2F%2Fwww.espaciolinux.com%2Fforos%2Fprogramacion%2Fcompilar-programa-linux-t11627.html&rct=j&q=programar%20en%20c%20linux&ei=dr7tTYyEPISitgfLv_2nCQ&usg=AFQjCNGBBtprxcyzfdPxLzdOmihcxoxKsA&sig2=JQmJEAwdT4uiHtxFlSQ57w&cad=rja

**10. MINDIASPORA, 2000**

http://www.mindiaspora.am/sp/pan_armenian_networks

# GLOSARIO DE TERMINOS

**FPGA:** (Field Programmable Gate Array)

**PAN:** Personal Area Network – red de área personal

**ZIGBEE:** Conjunto de protocolos de alto nivel de comunicación inalámbrica

**MICROCONTROLADOR**: Es la unidad central de procesamiento.

**RADIOFRECUENCIA**: También denominado espectro de radiofrecuencia o RF.,

**INFRARROJO:** Es una radiación electromagnética cuya frecuencia la hace invisible al ojo humano.

**TOPOLOGIA:** Se define como la cadena de comunicación usada por los nodos que conforman una red para comunicarse

**CIFRADO:** El cifrado es un método que permite aumentar la seguridad de un mensaje o de un archivo

**ENCRIPTACION:** Es el proceso para volver ilegible.

**ADC**: Conversión de analógica a digital

**VCC:** Voltaje de corriente directa

**GND:** Ground tierra

**MODULADOR:** Dispositivo electrónico

**KBPS:** Kilobites por segundo

**MAC:** Control de acceso al medio.

**PDA:** Personal Digital Assistant – Assistente digital personal

**WPAN:** Wireless Personal Area Networks, Red Inalámbrica de Área Persona.

**E/S:** Entrada Salida

**VIN:** Voltaje de entrada

**CPU:** Unidad central de proceso.

**USB:** Universal Serial Bus- bus serial universal.

# ANEXOS

# ANEXO 1

## Arquitectura de la FPGA Spartan III de Xilinx

Las FPGA Spartan III de Xilin están conformadas por un conjunto de Bloques Lógicos Configurables (Configurable Logic Blocks: CLBs) rodeados por un perímetro de Bloques Programables de entrada/salida (Programmable Input/Output Blocks: IOBs). Estos elementos funcionales están interconectados por una jerarquía de canales de conexión (Routing Channels), la que incluye una red de baja capacitancia para la distribución de señales de reloj de alta frecuencia. Adicionalmente el dispositivo cuenta con 24 bloques de memoria RAM de 2Kbytes de doble puerto, cuyos anchos de buses son configurables, y con 12 bloques de multiplicadores dedicados de 18 X 18 bits.

Los cinco elementos funcionales programables que la componen son los siguientes:

- Bloques de entrada/salida (Input/Output Blocks – IOBs): Controlan el flujo de datos entre los pines de entrada/salida y la lógica interna del dispositivo. Soportan flujo bidireccional más operación tri-estado y un conjunto de estándares de voltaje e impedancia controlados de manera digital.
- Bloques Lógicos configurables (Configurable Logic Blocks – CLBs): Contienen Look-Up Tables basadas en tecnología RAM (LUTs) para implementar funciones lógicas y elementos de almacenamiento que pueden ser usados como flip-flops o como latches.
- Bloques de memoria RAM (Block RAM): Proveen almacenamiento de datos en bloques de 18 Kbits con dos puertos independientes cada uno.
- Bloques de multiplicación que aceptan dos números binarios de 18 bit como entrada y entregan uno de 36 bits.
- Administradores digitales de reloj (Digital Clock Managers – DCMs): Estos elementos proveen funciones digitales auto calibradas, las que se encargan de distribuir, retrazar arbitrariamente en pocos grados, desfasar en 90, 180, y 270 grados, dividir y multiplicar las señales de reloj de todo el circuito.

Los elementos descritos están organizados como se muestra en la Figura 1. Un anillo de IOBs rodea un arreglo regular de CLBs. Atraviesa este arreglo una columna de Bloques de memoria RAM, compuesta por varios bloques de 18 Kbit, cada uno de los cuales está

asociado con un multiplicador dedicado. Los DCMs están colocados en los extremos de dichas columnas.

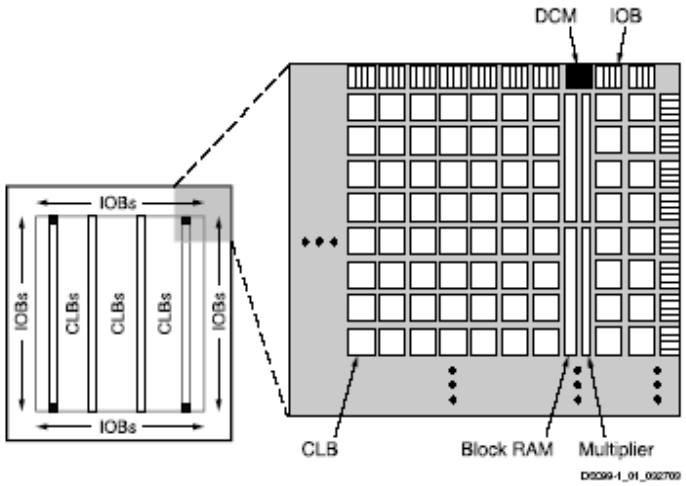



**Figura 1: Arquitectura de la Spartan III**

A continuación se hace una descripción más detallada de cada uno de los elementos funcionales de la FPGA, y luego se describe el proceso de configuración de la misma.

## Bloques de entrada/salida IOB

Los bloques de entrada/salida (IOB) suministran una interfaz bidireccional programable entre un pin de entrada/salida y la lógica interna de la FPGA. Un diagrama simplificado de la estructura interna de un IOB aparece en la Figura 2. Hay tres rutas para señales en un IOB: la ruta de salida, la ruta de entrada y la ruta tri-estado. Cada ruta tiene su propio par de elementos de almacenamiento que pueden actuar tanto como registros o como latches. Las tres rutas principales son como sigue:

- La ruta de entrada, que lleva datos desde el pad, que está unido al pin del package, a través de un elemento de retardo opcional programable, directamente a la línea I. Después del elemento de retardo hay rutas alternativas a través de un par de elementos de almacenamiento hacia las líneas IQ1 e IQ2. Las tres salidas del IOB todas conducen a la lógica interna de la FPGA.

- La ruta de salida, que parte con las líneas O1 y O2, lleva datos desde la lógica interna de la FPGA, a través de un multiplexor y del driver tri-estado hacia el pad del IOB. En suma a esta ruta directa, el multiplexor da la opción de insertar un par de elementos de almacenamiento.
- La ruta tri-estado determina cuando el driver de salida está en alta impedancia. Las líneas T1 y T2 llevan datos desde la lógica interna a través de un multiplexor hacia el driver de salida. En suma a esta ruta directa, el multiplexor da la opción de entregar un par de elementos de almacenamiento.

Todas las rutas de señales que entran al IOB, incluidas aquellas asociadas con los elementos de almacenamiento tienen una opción de inversión. Cualquier inversor colocado (en la programación) en estas rutas es automáticamente absorbido dentro del IOB.

Hay tres pares de elementos de almacenamiento en cada IOB, un par para cada uno de las tres rutas. Es posible configurar cada uno de esos elementos como un flip-flop D gatillado por flanco (FD) o como un latch sensible a nivel (LD). Estos elementos son controlados con la misma red de distribución de relojes que se utiliza para todo el sistema.

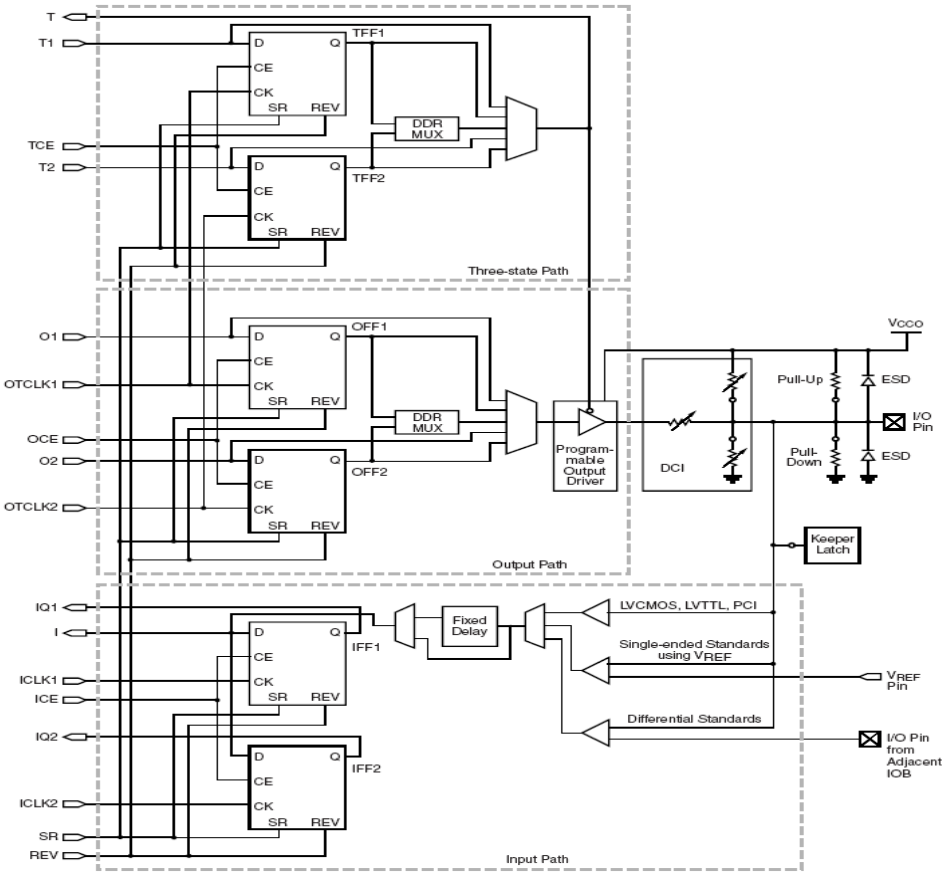



**Figura 2: Diagrama simplificado de un IOB de la Spartan III**

El par de elementos de almacenamiento tanto de la ruta de salida o de la del driver tri-estado pueden ser usados en conjunto, con un multiplexor especial para producir transmisión de doble tasa de datos (DDR). Esto se logra tomando datos sincronizados con el flanco de subida del reloj y convirtiéndolos en bits sincronizados tanto con el flanco de subida como con el de bajada. A esta combinación de dos registros y un multiplexor se le llama flip flop tipo D de doble tasa de datos (FDDR).

Cada IOB cuenta además con otros elementos, entre los cuales cuentan las resistencias de Pull-Up y de Pull-Down, que tienen el objetivo de establecer niveles altos o bajos respectivamente en las salidas de los IOBs que no están en uso; un circuito de retención (Keeper) del último nivel lógico que se mantiene, después de que todos los drivers han sido apagados, lo que es útil para cuidar que las líneas de un bus no floten, cuando los

drivers conectados están en alta impedancia; un circuito de protección para descargas electro estáticas (protección ESD), que utiliza diodos de protección.

Finalmente cada IOB cuenta con un control para el slew rate y para la corriente de salida máxima. El primero otorga la posibilidad de elegir una tasa alta de cambio de nivel (con bajo slew rate) o una tasa máxima menor, pero con un control de transciente, para la utilización de los puertos en la integración a buses, donde al pasar de alta impedancia a un nivel de voltaje suele producirse transiciones inesperadas. El segundo entrega siete niveles deferentes de corrientes máximas tanto para el estándar CMOS como para el TTL, lo que permite adaptarse a dispositivos que necesitan mayores corrientes para su activación; en el caso del estándar LVCMOS a 2.5V el rango de corrientes es de 2 a 24 mA.(2, 4, 6, 8, 12, 16, 24 mA).

Los IOB soportan 17 estándares de señales de salida de terminación única y seis de señal diferencial; también cuentan con un sistema integrado, para coincidir con la impedancia de las líneas de transmisión que llegan a la FPGA, llamado Control Digital de Impedancia (DCI), el que permite elegir hasta 5 tipos diferentes de terminaciones, utilizando una red de resistencias internas que se ajustan en serie o en paralelo, dependiendo de las necesidades del estándar elegido.

## Bloques de Lógica Configurable (CLB)

El bloque básico de la red que compone la FPGA es la slice. Existen dos tipo de slice, éstas se diferencian en algunos elementos, pero son muy parecidas, (ver más adelante). Luego estas slices se organizan en los bloques lógicos elementales, que son los que se describen a continuación.

Los Bloques de Lógica Configurable (CLBs) constituyen el recurso lógico principal para implementar circuitos síncronos o combinacionales. Cada CLB está compuesta de cuatro slices interconectadas entre si, tal como se muestra en la Fugura 3

Las cuatro slices que componen un CLB tienen los siguientes elementos en común: dos generadores de funciones lógicas, dos elementos de almacenamiento, multiplexores de función amplia, lógica de carry y compuertas aritméticas, tal como se muestra en la Figura . Los dos pares de slices usan estos elementos para entregar funciones lógicas y aritméticas de ROM. Además de lo anterior, el par de la izquierda soporta dos funciones adicionales: almacenamiento de datos usando RAM distribuida y corrimiento de datos con registros de 16 bits.

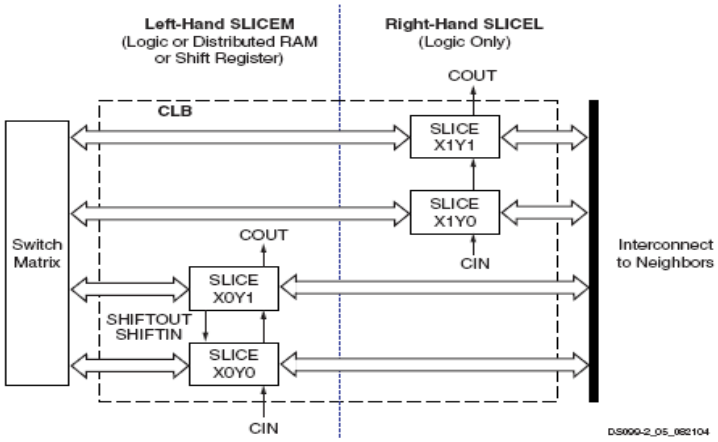



**Figura 3: Arreglo de slices en un CLB**

La Figura 3 es un diagrama de una slice del par del lado izquierdo, por lo tanto representa un súper conjunto de los elementos y conexiones que se encuentran en las slices.

El generador de funciones basado en RAM –también conocido como Look-Up Table (LUT)- es el recurso principal para implementar funciones lógicas dentro de la FPGA. Más aún, las LUTs en cada par de slices del lado izquierdo pueden ser configuradas como RAM distribuida o como un registro de corrimiento de 16 bits. Los generadores de funciones ubicados en las porciones superiores e inferiores de la slice son referidos como “G-LUT” y “F-LUT” respectivamente en la Figura 4.

El elemento de almacenamiento, el cual es programable tanto como un flip flop tipo D o como un latch sensible a nivel, provee un medio para sincronizar datos a una señal de

reloj, entre otros usos. Estos elementos de almacenamiento, que se encuentran en las porciones superiores e inferiores de la slice son llamados “FFY” y “FFX”, respectivamente.

Los multiplexores de función amplia combinan las LUTs para permitir operaciones lógicas más complejas, cada slice tiene dos de éstos, en la Figura 4 corresponden a F5MUX y F1MUX.

La cadena de carry, en combinación con varias compuertas lógicas dedicadas, soportan implementaciones rápidas de operaciones matemáticas. La cadena de carry entra a la slice como CIN y sale como COUT. Cinco multiplexores controlan la cadena: CYINIT, CY0F y CYMUXF en la porción inferior, así como CY0G y CYMUXG en la porción superior. La lógica aritmética dedicada incluye compuertas XOR y AND en cada porción de la slice.

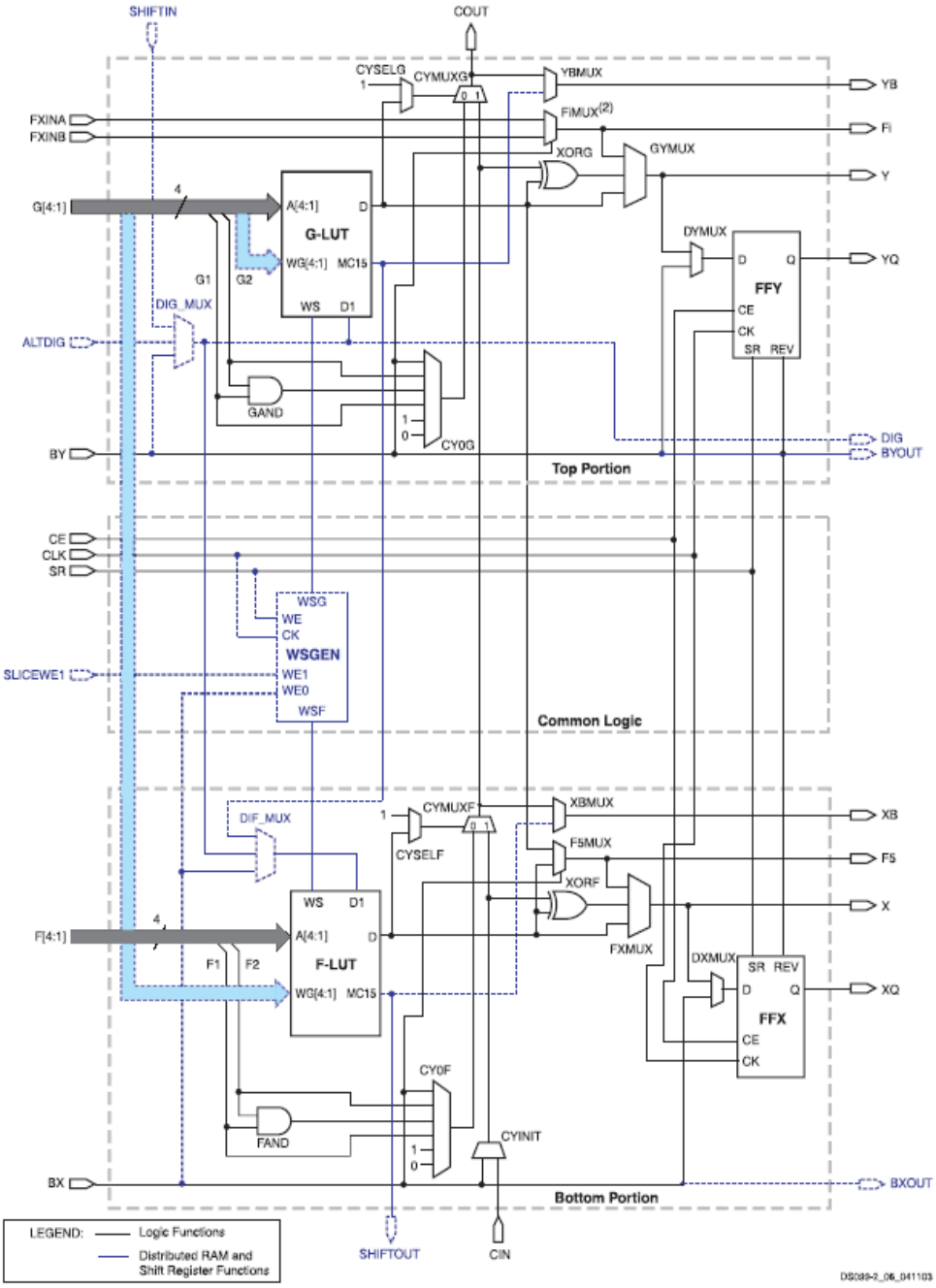



**Figura 4: Diagrama simplificado de una slice del lado izquierdo de un CLB**

Con un rol central en la operación de cada slice se encuentran dos rutas de datos casi idénticas. Para la descripción que prosigue se usan los nombres de la parte inferior de la Figura 4. A ruta básica tiene su origen en la matriz de switches de interconexión

colocada fuera del CLB. Cuatro líneas, F1 a F4 entran en la slice y se conectan directamente a la LUT. Una vez dentro de la slice, la ruta de los 4 bits inferiores pasa a través de un generador de funciones F que realiza operaciones lógicas. La ruta de salida del generador de funciones, D, ofrece cinco posibles rutas posibles:

- Salir de la slice por la línea X y volver a interconectarse.
- Dentro de la slice, X sirve como entrada al DXMUX que alimenta la entrada de datos D, correspondiente al elemento de almacenamiento FFY. La salida Q de este elemento maneja la ruta XQ que sale de la slice.
- Controlar el multiplexor CYMUXF de la cadena de carry.
- Con la cadena de carry, servir como una entrada a la compuerta XORF, que realiza operaciones aritméticas y produce el resultado en X.
- Manejar el multiplexor F5MUX para implementar funciones lógicas más anchas que 4 bits. Las salidas D de los F-LUT y G-LUT sirven de entradas de datos para este multiplexor.

En suma a los caminos lógicos principales descritos recién, existen dos rutas de bypass que entran a la slice como BX y BY. Una vez dentro de la FPGA, BX en la parte de debajo de la slice (o BY en la parte superior) puede tomar cualquiera de varias ramas diferentes:

- Hacer bypass de la LUT y del elemento de almacenamiento, luego salir de la slice como BXOUT y volver a interconectarse.
- Hacer bypass a la LUT, y luego pasar a través del elemento de almacenamiento, para luego salir como XQ.
- Controlar el multiplexor F5MUX.
- Servir como una entrada a la cadena de carry vía los multiplexores.
- Manejar la entrada DI de la LUT.
- BY puede controlar la entrada REV de FFY y de FFX.
- Finalmente, el multiplexor DIG_MUX puede derivar la ruta BY hacia la línea DIG que sale de la slice.

Cada una de las dos LUTs (F y G) de una slice tiene cuatro entradas lógicas (A1–A4) y una única salida D. Esto permite programar cualquier operación lógica booleana de cuatro variables en este dispositivo. Además, los multiplexores de función amplia pueden usarse para combinar LUTs dentro del mismo CLB o incluso a través de diferentes CLBs, haciendo posible funciones con mayor número de variables.

Las LUT de ambos pares de slices dentro de un CLB no sólo soportan las funciones descritas, si no que también pueden funcionar como ROM (Read Only Memory) con datos inicializados al momento de configurar la FPGA. Las LUTs del lado izquierdo de cada CLB soportan además dos funciones adicionales: primero, es posible programarlas como RAM distribuida, lo que permite contar con espacios de memoria de 16 bits en cualquier parte de la topología de la FPGA. Segundo, es posible programar una de estas LUTs como un registro de desplazamiento de 16 bits, con lo que se pueden producir retardos de hasta 16 bits o combinaciones de varias LUTs pueden producirlos de cualquier largo de bits.

## Bloques dedicados de memoria RAM

La Spartan III tiene 24 bloques de 18 Kbits de memoria RAM. El ancho del bus de datos versus el de direcciones (relación de aspecto) de cada bloque es configurable y se puede combinar varios de éstos para formar memorias más anchas o de mayor profundidad.

Tal como se muestra en la Figura 5, los bloques de RAM tienen una estructura de doble puerto. Dos puertos idénticos llamados A y B permiten acceso independiente al mismo rango de memoria, que tiene una capacidad máxima de 18.432 bits – o 16.384 cuando no se usan las líneas de paridad. Cada puerto tiene su propio set de líneas de control, de datos y de reloj para las operaciones síncronas de lectura y escritura. Estas operaciones tienen lugar de manera totalmente independiente en cada uno de los puertos.

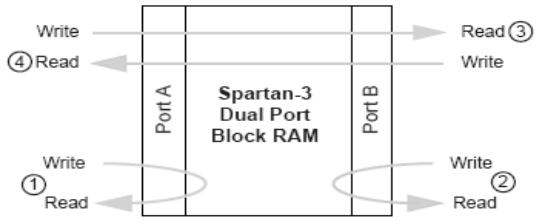



**Figura 5: Diagrama de un bloque de RAM dedicado de la Spartan III**

## Proceso de configuración de la FPGA Spartan III

La FPGA Spartan III se programa por medio de la carga de los datos de configuración en celdas de memoria estática, las que colectivamente controlan todos los elementos funcionales y los recursos de interconexión. Luego de aplicar alimentación a la oblea, se escribe la trama de configuración en dicha memoria utilizando uno de los siguientes modos: Maestro - Paralelo, Esclavo - Paralelo, Maestro - Serial, Esclavo - Serial o Boundary-Scan (JTAG). Estos modos difieren en el origen del reloj (proviene de la FPGA en los modos Maestro y es externo en los modos Esclavo), y en la forma en que se escriben los datos, por lo que los modos paralelos son más rápidos.

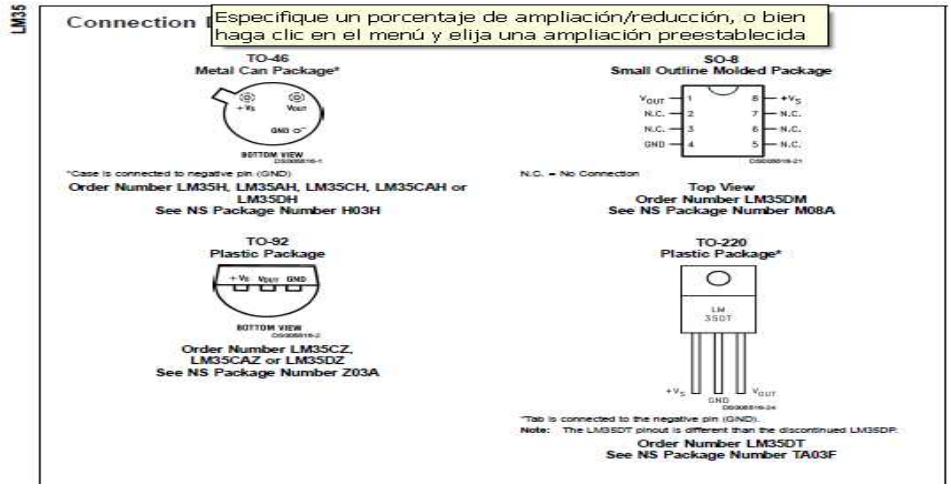
El modo Boundary-Scan utiliza pines dedicados de la FPGA y cumple con los estándares IEEE 1149.1 Test Access Port e IEEE 1532 para dispositivos In-System Configurable (ISC). Este modo está siempre disponible en la FPGA y al activarlo se desactivan los otros modos ya mencionados.

El proceso de configuración de la FPGA ocurre en tres etapas. Primero la memoria interna de configuración es borrada. Luego los datos de configuración son cargados en dicha memoria, y finalmente la lógica es activada por un proceso de partida.

# ANEXO 2

# DATASHEET

Conexiones en LM35DZ




Velocidad Máxima de LM35DZ

## Electrical Characteristics

(Notes 1, 6)

| Parameter | Conditions | LM35A | | | LM35CA | | | Units (Max.) |
|---|---|---|---|---|---|---|---|---|
| | | Typical | Tested Limit (Note 4) | Design Limit (Note 5) | Typical | Tested Limit (Note 4) | Design Limit (Note 5) | |
| Accuracy (Note 7) | $T_A$=+25°C | ±0.2 | ±0.5 | | ±0.2 | ±0.5 | | °C |
| | $T_A$=−10°C | ±0.3 | | | ±0.3 | | ±1.0 | °C |
| | $T_A$=$T_{MAX}$ | ±0.4 | ±1.0 | | ±0.4 | ±1.0 | | °C |
| | $T_A$=$T_{MIN}$ | ±0.4 | ±1.0 | | ±0.4 | | ±1.5 | °C |
| Nonlinearity (Note 8) | $T_{MIN}$≤$T_A$≤$T_{MAX}$ | ±0.18 | | ±0.35 | ±0.15 | | ±0.3 | °C |
| Sensor Gain (Average Slope) | $T_{MIN}$≤$T_A$≤$T_{MAX}$ | +10.0 | +9.9, +10.1 | | +10.0 | | +9.9, +10.1 | mV/°C |
| Load Regulation (Note 3) 0≤$I_L$≤1 mA | $T_A$=+25°C | ±0.4 | ±1.0 | | ±0.4 | ±1.0 | | mV/mA |
| | $T_{MIN}$≤$T_A$≤$T_{MAX}$ | ±0.5 | | ±3.0 | ±0.5 | | ±3.0 | mV/mA |
| Line Regulation (Note 3) | $T_A$=+25°C | ±0.01 | ±0.05 | | ±0.01 | ±0.05 | | mV/V |
| | 4V≤$V_S$≤30V | ±0.02 | | ±0.1 | ±0.02 | | ±0.1 | mV/V |
| Quiescent Current (Note 9) | $V_S$=+5V, +25°C | 56 | 67 | | 56 | 67 | | µA |
| | $V_S$=+5V | 105 | | 131 | 91 | | 114 | µA |
| | $V_S$=+30V, +25°C | 56.2 | 68 | | 56.2 | 68 | | µA |
| | $V_S$=+30V | 105.5 | | 133 | 91.5 | | 116 | µA |
| Change of Quiescent Current (Note 3) | 4V≤$V_S$≤30V, +25°C | 0.2 | 1.0 | | 0.2 | 1.0 | | µA |
| | 4V≤$V_S$≤30V | 0.5 | | 2.0 | 0.5 | | 2.0 | µA |
| Temperature Coefficient of Quiescent Current | | +0.39 | | +0.5 | +0.39 | | +0.5 | µA/°C |
| Minimum Temperature for Rated Accuracy | In circuit of *Figure 1*, $I_L$=0 | +1.5 | | +2.0 | +1.5 | | +2.0 | °C |
| Long Term Stability | $T_J$=$T_{MAX}$, for 1000 hours | ±0.08 | | | ±0.08 | | | °C |

# ANEXO 3

## Código de conexión UART

```
library ieee;
   use ieee.std_logic_1164.all;
    use ieee.std_logic_unsigned.all;
 entity uart is
   port (
       reset       :in  std_logic;
       txclk       :in  std_logic;
       ld_tx_data  :in  std_logic;
       tx_data     :in  std_logic_vector (7 downto 0);
       tx_enable   :in  std_logic;
       tx_out      :out std_logic;
       tx_empty    :out std_logic;
       rxclk       :in  std_logic;
       uld_rx_data :in  std_logic;
       rx_data     :out std_logic_vector (7 downto 0);
       rx_enable   :in  std_logic;
       rx_in       :in  std_logic;
       rx_empty    :out std_logic
    );
 end entity;
 architecture rtl of uart is
   -- Internal Variables
    signal tx_reg        :std_logic_vector (7 downto 0);
    signal tx_over_run   :std_logic;
    signal tx_cnt        :std_logic_vector (3 downto 0);
    signal rx_reg        :std_logic_vector (7 downto 0);
    signal rx_sample_cnt  :std_logic_vector (3 downto 0);
    signal rx_cnt        :std_logic_vector (3 downto 0);
    signal rx_frame_err   :std_logic;
    signal rx_over_run    :std_logic;
    signal rx_d1          :std_logic;
    signal rx_d2          :std_logic;
    signal rx_busy        :std_logic;
    signal rx_is_empty    :std_logic;
    signal tx_is_empty    :std_logic;
 begin
   -- UART RX Logic
    process (rxclk, reset) begin
       if (reset = '1') then
          rx_reg       <= (others=>'0');
          rx_data      <= (others=>'0');
          rx_sample_cnt <= (others=>'0');
          rx_cnt       <= (others=>'0');
          rx_frame_err  <= '0';
          rx_over_run   <= '0';
          rx_is_empty   <= '1';
          rx_d1         <= '1';
```

```
            rx_d2         <= '1';
            rx_busy       <= '0';
          elsif (rising_edge(rxclk)) then
            -- Synchronize the asynch signal
            rx_d1 <= rx_in;
            rx_d2 <= rx_d1;
            -- Uload the rx data
            if (uld_rx_data = '1') then
               rx_data  <= rx_reg;
               rx_is_empty <= '1';
            end if;
-- Receive data only when rx is enabled
if (rx_enable = '1') then
-- Check if just received start of frame
               if (rx_busy = '0' and rx_d2 = '0') then
                  rx_busy       <= '1';
                  rx_sample_cnt <= X"1";
                  rx_cnt        <= X"0";
               end if;
-- Start of frame detected, Proceed with rest of data
               if (rx_busy = '1') then
                  rx_sample_cnt <= rx_sample_cnt + 1;
                 -- Logic to sample at middle of data
                  if (rx_sample_cnt = 7) then
                     if ((rx_d2 = '1') and (rx_cnt = 0)) then
                        rx_busy <= '0';
                     else
                        rx_cnt <= rx_cnt + 1;
                       -- Start storing the rx data
                        if (rx_cnt > 0 and rx_cnt < 9) then
                           rx_reg(conv_integer(rx_cnt) - 1) <= rx_d2;
                     end if;       if (rx_cnt = 9) then
                           rx_busy <= '0';
                          -- Check if End of frame received correctly
                           if (rx_d2 = '0') then
                              rx_frame_err <= '1';
                           else
                              rx_is_empty  <= '0';
                              rx_frame_err <= '0';
                             -- Check if last rx data was not unloaded,
                              if (rx_is_empty = '1') then
                                 rx_over_run  <= '0';
                              else
                                 rx_over_run  <= '1';
                              end if;
                           end if;
                     end if;
```

```
                    end if;
                  end if;
               end if;
            end if;
          if (rx_enable = '0') then
               rx_busy <= '0';
            end if;
          end if;
      end process;
      rx_empty <= rx_is_empty;

     -- UART TX Logic
      process (txclk, reset) begin
         if (reset = '1') then
            tx_reg       <= (others=>'0');
            tx_is_empty   <= '1';
            tx_over_run   <= '0';
            tx_out       <= '1';
            tx_cnt       <= (others=>'0');
         elsif (rising_edge(txclk)) then

if (ld_tx_data = '1') then
if (tx_is_empty = '0') then
          tx_over_run <= '0';
               else
           tx_reg   <= tx_data;
           tx_is_empty <= '0';
               end if;
            end if;
if (tx_enable = '1' and tx_is_empty = '0') then
               tx_cnt <= tx_cnt + 1;
               if (tx_cnt = 0) then
                  tx_out <= '0';
               end if;
               if (tx_cnt > 0 and tx_cnt < 9) then
                  tx_out <= tx_reg(conv_integer(tx_cnt) -1);
               end if;
               if (tx_cnt = 9) then
                  tx_out <= '1';
                  tx_cnt <= X"0";
                  tx_is_empty <= '1';
               end if;
            end if;
            if (tx_enable = '0') then
               tx_cnt <= X"0";
            end if;
         end if;
```

```
    end process;
    tx_empty <= tx_is_empty;

end architecture;
```

# ANEXO 4

## Conexión ADC

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;
entity ADC_8_bit is
  port (analog_in : in real range -15.0 to +15.0;
        digital_out : out std_logic_vector(7 downto 0)
       );
end entity;
architecture original of ADC_8_bit is
  constant conversion_time: time := 25 ns;
  signal instantly_digitized_signal : std_logic_vector(7 downto 0);
  signal delayed_digitized_signal : std_logic_vector(7 downto 0);
  function ADC_8b_10v_bipolar (
    analog_in: real range -15.0 to +15.0
  ) return std_logic_vector is
    constant max_abs_digital_value :
integer := 128;
    constant max_in_signal : real := 10.0;
    variable analog_signal: real;
    variable analog_abs: real;
    variable analog_limited: real;
    variable digitized_signal: integer;
    variable digital_out: std_logic_vector(7 downto 0);
  begin
analog_signal := real(analog_in);
```

```
    if (analog_signal < 0.0) then    -- i/p = -ve
      digitized_signal := integer(analog_signal * 12.8);
      if (digitized_signal < -(max_abs_digital_value)) then
        digitized_signal := -(max_abs_digital_value);
      end if;
    else    -- i/p = +ve
      digitized_signal := integer(analog_signal * 12.8);
      if (digitized_signal > (max_abs_digital_value - 1)) then
        digitized_signal := max_abs_digital_value - 1;
      end if;
    end if;
    digital_out := std_logic_vector(to_signed(digitized_signal, digital_out'length));
    return digital_out;
  end ADC_8b_10v_bipolar;
begin
  s0: instantly_digitized_signal <=
        std_logic_vector (ADC_8b_10v_bipolar (analog_in));
  s1: delayed_digitized_signal <=
        instantly_digitized_signal after conversion_time;
  s2: digital_out <= delayed_digitized_signal;
end original;
```

# ANEXO 5

Listado del código para la interfaz gráfica

```
namespace XBEE
{
    public partial class Form1 : Form
    {
        public Bitmap image1;
        public string dato;
        public int hacer = 0;
        public string temp;
        public Pen fini = new Pen(Color.Red, 1);
        public Pen lim = new Pen(Color.White, 1);
        public Bitmap lienzo = new Bitmap(1500, 1500);
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        //    serialPort1.Open();
            Form2 fo = new Form2();
            fo.Close();

        }

        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            dato = serialPort1.ReadExisting();
            this.Invoke(new EventHandler(lecturas));
        }
        private void lecturas(object sender, EventArgs e)
        {
                textBox1.AppendText(dato);
                timer1.Enabled = true;
                temp = textBox1.Text;
                hacer = 1;
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            timer1.Enabled = false;
            int a;
            string dato1;
            string dato2;
            string dato3;
            textBox2.Text = temp.Substring(8, 4);
            a = temp.IndexOf("H");
            textBox4.Text = temp.Substring(14, a-14);
            textBox3.Text = temp.Substring(20,5);
            textBox5.Text = temp.Substring(26, 3);
            textBox6.Text = temp.Substring(0, 1);
            textBox7.Text = temp.Substring(1, 1);
            textBox8.Text = temp.Substring(2, 1);
            dato1 = textBox7.Text;
            dato2 = textBox6.Text;
            dato3 = textBox8.Text;
            if (dato1 == "D")
```

```
        {
            image1 = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(image1);
            g.DrawRectangle(fini, 20, 20, 300, 325);

        }
        else
        {
            image1 = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(image1);
            g.DrawRectangle(lim, 20, 20, 300, 325);
        }
        if (dato2 == "R")
        {
            image1 = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(image1);
            g.DrawRectangle(fini, 350, 20, 430, 325);
        }
        else
        {
            image1 = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(image1);
            g.DrawRectangle(lim, 350, 20, 430, 325);
        }

        if (dato3 == "P")
        {
            image1 = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(image1);
            g.DrawRectangle(fini, 20, 370, 758, 268);
        }
        else
        {
            image1 = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(image1);
            g.DrawRectangle(lim, 20, 370, 758, 268);
        }
        this.pictureBox1.Image = image1;
        this.pictureBox1.Refresh();
        textBox1.Text = "";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        serialPort1.Write("E");
    }

    private void button2_Click(object sender, EventArgs e)
    {
        serialPort1.Write("A");
    }

    private void button3_Click(object sender, EventArgs e)
    {
        image1 = (Bitmap)pictureBox1.Image;
        Graphics g = Graphics.FromImage(image1);
        g.DrawRectangle(fini, 30, 30, 100, 100);
```

```
            this.pictureBox1.Image = image1;
            this.pictureBox1.Refresh();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            serialPort1.Write("O");
        }

        private void button4_Click(object sender, EventArgs e)
        {
            serialPort1.Write("N");
        }

        private void button9_Click(object sender, EventArgs e)
        {
            serialPort1.Write("I");
        }

        private void button8_Click(object sender, EventArgs e)
        {
            serialPort1.Write("D");
        }

        private void button11_Click(object sender, EventArgs e)
        {
            serialPort1.Write("M");
        }

        private void button10_Click(object sender, EventArgs e)
        {
            serialPort1.Write("L");
        }

        private void button12_Click(object sender, EventArgs e)
        {

        }

        private void button12_Click_1(object sender, EventArgs e)
        {
            string a;
            a = comboBox1.Text;
            if (a == "")
                MessageBox.Show("SELECCIONE PUERTO");
            else
            {
                serialPort1.PortName = a;
                serialPort1.Open();
            }
        }

        void PictureBox1Click(object sender, System.EventArgs e)
        {

         }
}}
```

```
namespace XBEE
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string a,b;
            a = textBox1.Text;
            b = textBox2.Text;
            if ((a == "Administrator") && (b == "UPS"))
            {

                Form1 form = new Form1();
                form.ShowDialog();
                textBox1.Text = "";
                textBox2.Text = "";

            }
            else
            {
                MessageBox.Show("Error Usuario");
                textBox1.Text = "";
                textBox2.Text = "";
            }


        }

        void TextBox1TextChanged(object sender, System.EventArgs e)
        {

         }
    }
}
```

# ANEXO 6

# CÓDIGO SIMPLIFICADO DEL UART PARA EL FPGA

```
------------------------------------------------------------
-- Design Name : uart
-- File Name   : uart.vhd
-- Function    : Tx  - RxUART
-- NAme        : Darwin Castillo - Richard Osorio
-- University  : UPS
------------------------------------------------------------
library ieee;
   use ieee.std_logic_1164.all;
   use ieee.std_logic_unsigned.all;

entity uart is
   port (
      reset      :in  std_logic;
      txclk      :in  std_logic;
      ld_tx_data  :in  std_logic;
      tx_data     :in  std_logic_vector (7 downto 0);
      tx_enable   :in  std_logic;
      tx_out      :out std_logic;
      tx_empty    :out std_logic;
      rxclk       :in  std_logic;
      uld_rx_data :in  std_logic;
      rx_data     :out std_logic_vector (7 downto 0);
      rx_enable   :in  std_logic;
      rx_in       :in  std_logic;
      rx_empty    :out std_logic
```

```
    );
end entity;
architecture rtl of uart is
    -- Variables Internas
    signal tx_reg        :std_logic_vector (7 downto 0);
    signal tx_over_run    :std_logic;
    signal tx_cnt        :std_logic_vector (3 downto 0);
    signal rx_reg        :std_logic_vector (7 downto 0);
    signal rx_sample_cnt  :std_logic_vector (3 downto 0);
    signal rx_cnt        :std_logic_vector (3 downto 0);
    signal rx_frame_err   :std_logic;
    signal rx_over_run    :std_logic;
    signal rx_d1         :std_logic;
    signal rx_d2         :std_logic;
    signal rx_busy       :std_logic;
    signal rx_is_empty    :std_logic;
    signal tx_is_empty    :std_logic;
begin
    -- Logica de Rx UART
    process (rxclk, reset) begin
        if (reset = '1') then
            rx_reg       <= (others=>'0');
            rx_data      <= (others=>'0');
            rx_sample_cnt <= (others=>'0');
            rx_cnt       <= (others=>'0');
            rx_frame_err  <= '0';
```

```
    rx_over_run   <= '0';
    rx_is_empty   <= '1';
    rx_d1         <= '1';
    rx_d2         <= '1';
    rx_busy       <= '0';
  elsif (rising_edge(rxclk)) then
    -- Sincronizacion de la senal asincronica
    rx_d1 <= rx_in;
    rx_d2 <= rx_d1;
    -- Descargando datos de rx
    if (uld_rx_data = '1') then
      rx_data  <= rx_reg;
      rx_is_empty <= '1';
    end if;
    -- Recibe datos solamente cuando rx esta habilitado
    if (rx_enable = '1') then
      -- Checa justo si esta recibiendo el inicio de la trama
      if (rx_busy = '0' and rx_d2 = '0') then
        rx_busy      <= '1';
        rx_sample_cnt <= X"1";
        rx_cnt       <= X"0";
      end if;
      -- Inicio del marco detectado, procediendo con el resto de datos
      if (rx_busy = '1') then
        rx_sample_cnt <= rx_sample_cnt + 1;
        -- La logica de muestra en promedio de datos
```

```
if (rx_sample_cnt = 7) then
   if ((rx_d2 = '1') and (rx_cnt = 0)) then
      rx_busy <= '0';
   else
      rx_cnt <= rx_cnt + 1;
      -- inicia almacenamiento de datos de rx
      if (rx_cnt > 0 and rx_cnt < 9) then
         rx_reg(conv_integer(rx_cnt) - 1) <= rx_d2;
      end if;
      if (rx_cnt = 9) then
         rx_busy <= '0';
         -- Checa si ha terminado de recibir la trama correctamente
         if (rx_d2 = '0') then
            rx_frame_err <= '1';
         else
            rx_is_empty  <= '0';
            rx_frame_err <= '0';
            -- Checa si el ultimo dato de rx no fue descargado,
            if (rx_is_empty = '1') then
               rx_over_run  <= '0';
            else
               rx_over_run  <= '1';
            end if;
         end if;
      end if;
   end if;
```

```
            end if;
          end if;
        end if;
        if (rx_enable = '0') then
          rx_busy <= '0';
        end if;
    end if;
end process;
rx_empty <= rx_is_empty;

-- Logica de Tx UART
process (txclk, reset) begin
    if (reset = '1') then
        tx_reg       <= (others=>'0');
        tx_is_empty   <= '1';
        tx_over_run   <= '0';
        tx_out       <= '1';
        tx_cnt       <= (others=>'0');
    elsif (rising_edge(txclk)) then

        if (ld_tx_data = '1') then
          if (tx_is_empty = '0') then
            tx_over_run <= '0';
          else
            tx_reg   <= tx_data;
            tx_is_empty <= '0';
```

```
            end if;
        end if;
        if (tx_enable = '1' and tx_is_empty = '0') then
            tx_cnt <= tx_cnt + 1;
            if (tx_cnt = 0) then
                tx_out <= '0';
            end if;
            if (tx_cnt > 0 and tx_cnt < 9) then
                tx_out <= tx_reg(conv_integer(tx_cnt) -1);
            end if;
            if (tx_cnt = 9) then
                tx_out <= '1';
                tx_cnt <= X"0";
                tx_is_empty <= '1';
            end if;
        end if;
        if (tx_enable = '0') then
            tx_cnt <= X"0";
        end if;
      end if;
   end process;
   tx_empty <= tx_is_empty;

end architecture;
```